Distributed Estimation and Control of Interacting Hybrid Systems for Traffic Applications

Gedistribueerde schatting en regeling
van interagerende hybride systemen in verkeerstoepassingen

Nicolae-Emanuel Marinică

Promotoren: prof. dr. ir. A. Sarlette, ereprof. dr. ir. R. Boel
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Werktuigkunde-Elektrotechniek

Vakgroep Elektrische Energie, Systemen en Automatisering
Voorzitter: prof. dr. ir. J. Melkebeek
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2013 - 2014

UNIVERSITEIT
GENT

ISBN 978-90-8578-662-7
NUR 950
Wettelijk depot: D/2014/10.500/8

Dit proefschrift is goedgekeurd door de promotoren:
Ereprof. dr. ir. René Boel
Prof. dr. ir. Alain Sarlette

Samenstelling promotiecommisie:

Prof. dr. ir. Luc Taerwe (prodecaan)	voorzitter
Ereprof. dr. ir. René Boel	Universiteit Gent, promoter
Prof. dr. ir. Alain Sarlette	Universiteit Gent, promoter
Prof. dr. ir. Dirk Aeyels	Universiteit Gent
Prof. dr. ir. El-Houssaine Aghezzaf	Universiteit Gent
Dr. Tom Maertens	Universiteit Gent
Prof. dr. ir. Jan van Schuppen	Technische Universiteit Delft, Delft, Nederland and Van Schuppen Control Research, Amsterdam, Nederland
Prof. dr. ir. Bart De Schutter	Technische Universiteit Delft, Delft, Nederland

Printed in Belgium
Authors email: nmarinica@gmail.com

Acknowledgements

This thesis is the result of several hard working years, very intense and full of ups and downs. I would like to express my gratitude to Prof. Dr. Rene Boel and Prof. Dr. Alain Sarlette, my supervisors. First, I would like to thank Rene for giving me freedom in research, the necessary support, and the countless debates that we had. Likewise, I also want to thank Alain for his support and dedication towards the finalization of this thesis.

I wish to acknowledge that the research presented in this thesis was made possible thanks to the Control for Coordination of Distributed Systems (CON4COORD ICT-2007.3.7.(c)) - EU FP7 project. The CON4COORD meetings and especially the people involved in this project helped to continuously improve myself. Special thanks to Prof. Dr. Jan van Schuppen for the challenging environment created within CON4COORD and later for his recommendations and interesting discussions during my visit at TUDelft. I am very grateful to the committee members for reading and helping me improve this thesis. I want to thank Prof. Dr. Ir. Bart de Schutter for allowing me to spend five fruitful months in the welcoming environment of DCSC at the Delft University of Technology. I would also like to thank Prof. Dr. Ir. Hans Hellendoorn which through his spontaneity and remarks made me feel welcomed. In Delft, it was also nice to socialise with Andrei, Ruxandra, Aleksandar, Elisabeth, and Arturo.

I would also like to thank Margot and Mia for their moral support and kind words during my wandering periods. Many thanks also to Gert who always succeeded to put a smile on my face and make me laugh no matter what my state of mind was at that point. For being friendly and creating a pleasant working atmosphere within SYSTeMS group, I would like to thank Filip, Fritz, Erik, Keivan, Rolando, Arthur, Jasper, Stavros, Bram, Raiku as well

as Nathan Huntley (UK) and Michael Kearny (AU). On a more personal note, many thanks to Filip for the many evenings spent together in front of a beer, discussing and sharing moments. Also, I would like to thank Herman, my office roommate, for the useful discussions and also for teaching me to have more patience.

During the time spent in Gent I met several people that I would also like to mention and thank for the nice moments spent together or their help: Luiza and Julien, Diana, Diana (G.), Timi (Ionel Craciun), Gabi, Dana, Cosmin , Miki, etc. (the list could go on and therefore I would like to be apologetic about missing someone from it). For their help and support, I want to thank to my landlady, Rita, and her son Peter. I am particularly grateful to Prof. Dr. Mircea Bodea for the inspiring discussions, moral guidance and corrections regarding real world perceptions.

On an even more personal note, I would like to thank my parents, Mariana and Mircea, for their support, encouragements, and for simply being there for me. Special thanks to my brother Adrian for always being ready to help me and for seeing the good side in everything. Finally, I want to thank my beloved wife, Raluca, for all her love, understanding, and patience with me. Without her support and cold judgements, I would not have made it. Thank you and sorry for not taking your advice ...

Nicolae Emanuel Marinică

Gent, February 5, 2014

Contents

Contents v

List of Figures ix

List of Symbols xv

1 Introduction 1

1.1 Overview of interacting hybrid systems 1

1.2 Objective and contributions 2

1.3 Thesis structure 6

2 Hybrid models for complex interacting systems 10

2.1 Discrete event dynamic systems 10

2.1.1 Automata 13

2.1.2 Timed automata 17

2.1.3 Timed input/output automata 19

2.2 Hybrid systems 20

2.2.1 Hybrid automata 21

2.2.2 World automata 24

2.3 Discrete-event simulations 25

2.4 Summary 26

3 Modelling urban traffic networks 27

3.1 Introduction 28

3.1.1 Link modelling 28

3.1.1.1 Transient behaviour 30

3.1.2 Intersection modelling 30
3.1.3 Urban traffic model classification 32
3.2 Real measurements available for the traffic analysis 33
3.3 Platoons in urban traffic 34
3.3.1 Experimental validation of platoon based vehicle aggregation 35
3.4 The platoon based model 39
3.4.1 Road section 40
3.4.2 Source 41
3.4.3 Sink 42
3.4.4 Intersection 42
3.4.5 Hierarchical model 43
3.4.6 Hybrid model 45
3.5 Implementation and performance 48
3.6 Summary 51

4 Distributed estimation of urban traffic networks 52
4.1 Issues for estimation 52
4.1.1 Sensor model 54
4.1.2 Arguments for using a particle filter 55
4.2 Standard particle filter 56
4.3 Distributed particle filter 61
4.4 Validation of the PF estimators 66
4.4.1 Centralised PF with the PBM as ground truth model 67
4.4.2 Centralised PF with SUMO as ground truth model 68
4.4.3 Distributed PF with SUMO as ground truth model 69
4.4.4 Histograms of the distributed PF with SUMO as ground truth model 70
4.5 Summary 74

5 Distributed feedback control of urban traffic networks with leader-follower coordination 75
5.1 Control of traffic 76

5.1.1 Literature context . . . 80
5.1.2 Platoons in control . . . 84
5.2 Preliminary control analysis . . . 86
5.2.1 A default switching rule for one saturated intersection . . . 87
5.2.2 A default switching rule for one intersection under very light traffic . . . 90
5.2.3 A default switching rule for one intersection under intermediate traffic . . . 91
5.3 Distributed feedback control with leader-follower coordination framework for urban traffic networks . . . 92
5.3.1 Control of a single intersection . . . 93
5.3.1.1 The optimisation cost C_o . . . 95
5.3.1.2 The final cost C_f . . . 97
5.3.1.3 Optimisation cost case study . . . 98
5.3.1.4 Solving the optimisation problem . . . 102
5.3.2 The not-following cost C_{nf} . . . 106
5.3.2.1 Leader-Follower control algorithm . . . 115
5.4 Validation examples . . . 117
5.4.1 Manhattan grid type network . . . 118
5.4.2 Non-Manhattan grid type network . . . 120
5.5 More realistic examples . . . 123
5.5.1 Manhattan grid type network with noise on the links . . . 124
5.5.2 Manhattan grid type network with right-turning traffic . . . 125
5.5.3 Non-Manhattan grid type network with right-turning traffic . . . 128
5.6 Summary . . . 129

6 Modelling and distributed control of the autonomous vehicles in a container terminal 131
6.1 Description of the system . . . 132
6.1.1 System architecture . . . 133
6.1.2 Autonomous vehicle specifications . . . 137
6.1.3 Controller specifications . . . 140
6.2 Control problem . . . 140

6.2.1 Successful communication case . . . 142
6.2.2 Unsuccessful communication case . . . 143
6.2.3 Manoeuvres . . . 144
6.3 Model of the case study . . . 146
6.3.1 Straddle carrier . . . 146
6.3.2 Environment . . . 148
6.3.3 Estimator . . . 149
6.3.4 Controller . . . 151
6.4 Validation examples . . . 151
6.4.1 Validation results . . . 153
6.5 Summary . . . 157

7 Conclusions and future work **158**
7.1 Conclusions and observations . . . 158
7.2 Future work . . . 164
7.2.1 Modelling of urban traffic networks . . . 164
7.2.2 Distributed estimation of urban traffic networks . . . 164
7.2.3 Distributed control of urban traffic networks . . . 166
7.2.4 Modelling and control for collision avoidance of autonomous vehicles . . . 167
7.3 Summary of the original contributions of the thesis . . . 168

References **171**

List of Figures

2.1 The automaton of a queueing system with limited capacity C. 15
2.2 The automata of 2 competing queueing systems with limited capacities $C1$ and $C2$ and the actuator Act. 16
2.3 Example with two queues in series connection modelled using the TIOA framework 20
2.4 The hybrid automaton of a queueing system with limited capacity C. . 22

3.1 Approximation of the macroscopic fundamental diagram - v_f called free velocity and represents the maximum legal velocity on a free lane. 29
3.2 Fluid flow model of intersection I_j 31
3.3 Area of Dendermonde between E17 - N17 where the measurements are taken (network indicated by the red line, signalised intersections indicated by rectangles, and area on which we focused encircled). . . 33
3.4 The traffic network in Dendermonde area. 34
3.5 Number of platoons (defined with maximal inter-vehicle time $\Delta = 5[\text{sec}]$) during the working days of 1 week. 35
3.6 Average size of the platoons (defined with maximal inter-vehicle time $\Delta = 5[\text{sec}]$) vs. traffic intensity for one working day with 2 hours step. 36
3.7 Inter-platoon time gap distribution (defined with maximal inter-vehicle time $\Delta = 5[\text{sec}]$) measured over 2 hours. 37
3.8 Components of an urban traffic network. 39
3.9 Consecutive platoons $P_{A_m,n-1}$ and $P_{A_m,n}$ emerging at the location A_m 42
3.10 The timed I/O automaton describing a queue: red states have $exit_i = 0$ (outflow blocked) and green states $exit_i = 1$ (outflow open), while for x_1 the condition of $exit_i$ does not matter. 45

3.11 The timed input/output automaton describing the evolution of a traffic light. 47
3.12 Structure diagram - dotted line represents the objects used the next chapters for estimation and control. 48
3.13 The queues formed at a typical intersection. The queues are in red and the phases of the traffic light are presented in green (sampling error due to the approximation made by the sensors). 50

4.1 Cumulative flow at two consecutive sensor locations on the same road. For a better comparison, the curve of the second sensor is shifted in time according to the estimated time delay $=$ (distance ≈ 1 [km] divided by mean-vehicle-speed ≈ 70 [km/h]) between the two sensors. 53
4.2 A small urban traffic network and its partitions for the DPF 63
4.3 The road topology and the sensor's locations for the validation study . 66
4.4 Estimated queue (pink dashed line) vs. real queue (full red line) generated by the PBM, in presence of a sudden accident 68
4.5 Queue estimated by the PBM-based central PF (pink dashed line) vs. real queue generated by SUMO (full red line) 69

5.1 Five intersections interconnected in a star topology. 86
5.2 Cases of the local cost. 97
5.3 The penalisation of the queue that is not zero after the T_h. 98
5.4 Switching time at T_a. 99
5.5 Switching time at T_b. 99
5.6 Switching time at T_c. 100
5.7 The cost can be convex or not depending on platoon arrival times. . . 101
5.8 Computation of the t_x time when the queue goes to 0 107
5.9 Average traffic during a nominal green period G_N 107
5.10 For $\eta = 0$ the follower is in phase with its leader and therefore there is not accumulated queue at the leader. 108
5.11 Case 2: if $0 < \eta < \frac{\lambda G_N}{\mu - \lambda}$ (for $\lambda \leq \left(1 - \frac{\sqrt{2}}{2}\right)\mu$) or $0 < \eta < G_N\left(1 - \frac{2\lambda}{\mu}\right)$ (for $\lambda > \left(1 - \frac{\sqrt{2}}{2}\right)\mu$) 109
5.12 Case 3a: if $\frac{\lambda G_N}{\mu - \lambda} < \eta < G_N\left(1 - \frac{2\lambda}{\mu}\right)$ (for $\lambda \leq \left(1 - \frac{\sqrt{2}}{2}\right)\mu$) 110

5.13 Case 3b: if $G_N\left(1-\frac{2\lambda}{\mu}\right) < \eta < \frac{\lambda G_N}{\mu-\lambda}$ (for $\lambda > \left(1-\frac{\sqrt{2}}{2}\right)\mu$) 110
5.14 Case 4: if $G_N\left(1-\frac{2\lambda}{\mu}\right) < \eta < G_N$ (for $\lambda \leq \left(1-\frac{\sqrt{2}}{2}\right)\mu$) or $\frac{\lambda G_N}{\mu-\lambda} < \eta < G_N$ (for $\lambda > \left(1-\frac{\sqrt{2}}{2}\right)\mu$) 111
5.15 Case 5: if $G_N < \eta < G_N + \frac{G_N\lambda}{\mu-\lambda}$ 112
5.16 Case 6: if $\frac{G_N\mu}{\mu-\lambda} < \eta < 2G_N$ 113
5.17 The $C_{nf}(\eta)$ for $\mu = 0.5,\ G_N = 32$, and $\lambda_1 = 0.2,\ \lambda_2 = 0.1$ 115
5.18 The phases of the 5 intersections using the leader-follower framework for $\omega = 0$. 119
5.19 The phases of the 5 intersections using the leader-follower framework for $\omega = 100$. 120
5.20 The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for Manhattan grid type network without noise. The performance of the fully distributed controller equivalent to $\omega = 0$ is represented in dashed line. 121
5.21 The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for Non-Manhattan grid type network without noise (in dashed line the performance of the fully distributed controller equivalent with $\omega = 0$). 122
5.22 The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for a Manhattan grid type network with noise on the links (red lines represent the median values of the data set and the blue horizonal line that goes through the whole graph represents the median for the network using the fully distributed approach, whose value is also represented by the rightmost bar-plots). 125
5.23 The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for a Manhattan grid type network with noise on the links (red lines represent the median values of the data set and the blue horizonal line that goes through the whole graph represents the median for the network using the fully distributed approach, whose value is also represented by the rightmost bar-plots) and right turning traffic. 126

5.24 The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for a Non - Manhattan grid type network with noise on the links (red lines represent the median values of the data set and the blue horizonal line that goes through the whole graph represents the median for the network using the fully distributed approach, whose value is also represented by the rightmost bar-plots) and right turning traffic. 127
5.25 A network with 4 leader agents (coloured in yellow), 15 follower agents (coloured in silver, one follower has two leaders) and 7 agents (neither leaders or followers) that do not have measurements and apply some nominal green cycles. 129

6.1 Schematic representation of the port quay of Antwerp. The stripes represent the row of containers, and the circles represent the straddle carriers. The row of cranes placed along the quay on the water side is not represented. 133
6.2 Architecture of the system. Note that the controllers are physically located in AVs (e.g. CA_N in AV_N). 135
6.3 AV_i in a possible collision with AV_j 136
6.4 Adjusting the path by adding vertices for two possible scenarios (dashed line is the current reference trajectory and full line is the new adjusted path). 145
6.5 Autonomous Vehicle world automaton. 147
6.6 Environment world automaton. 149
6.7 Estimator world automaton. 150
6.8 Controller world automaton. 152
6.9 Scenario with 3 AVs without communication 154
6.10 The delay histograms of the first example 154
6.11 Scenario with 4 AVs without communication 156
6.12 The delay histograms of the second example 156

List of Symbols

$\downarrow G^d_{j,c}$ The start time of the c-th green phase given for the direction d at intersection j

$\lambda(x,t)$ Flow, number of vehicles per unit of time at time t at location x on a link

$\lambda^j_i(t)$ Arrival flow, number of arrived vehicles per time unit at input i of intersection j

$\mathbb{F}_{\aleph}$ The set of followers for network $\aleph$

$\mathbb{F}^j_{\aleph}$ The followers set assigned to leader j

$\mathbb{L}_{\aleph}$ The set of leaders for network $\aleph$

$\mathbb{S}_{\aleph}$ The supervisor of network $\aleph$

$\mu_j(t)$ Departure flow, number of departing vehicles per unit of time at time t at intersection j

$\overline{G_j}$ Maximum green phase duration at intersection j

$\rho(x,t)$ Density, number of vehicles per unit area at time t at location x on a link

$\underline{G_j}$ Minimum green phase duration at intersection j

$\uparrow G^d_{j,c}$ The end time of the c-th green phase given for the direction d at intersection j

$\updownarrow G^d_{j,c}$ The time duration of the c-th green phase given for the direction d at intersection j

C_f The final cost.

C_{nf} The not-following cost paid by the followers for deviating from the leader's schedule

C_o The local optimisation cost.

CA_j Control agent j

$J_j\left(t_0, \uparrow G^d_{j,0}, \ldots, \uparrow G^d_{j,N_c-1}\right)$ The local cost criterion used by the control agent j

S_j The set of all possible switching scenarios

T_h The time horizon over which an optimal strategy is searched

$u_j(t)$ Open loop control strategy at intersection j at time t

$v\left(t\right)$ Velocity vector of a generic vehicle at time t

$v\left(x,t\right)$ Average speed at time t at location x on a link

$x\left(t\right)$ The position of a generic vehicle at time t

$S_{j,\circledast}(t_0)$ The optimal switching scenario of the next switching times at time t_0 for intersection j.

DEDS Discrete event dynamic systems

DES Discrete event simulation

ELCP Extended Linear Complementarity Problem

HIOA Hybrid input/output automata

OPAC Optimised Policies for Adaptive Control

PBM Platoon Based Model

RHODES Real-time Hierarchical, Optimised, Distributed, Effective System

SCATS Sydney Coordinated Adaptive Traffic System

SCOOT Split Cycle Offset Optimisation Technique

SUMO Simulation for Urban MObility

TIOA Timed Input/Output Automaton

WA World automata

Abstract

The distributed control of interacting hybrid systems presents difficulties because of (i) the distributed character implying only local observations of the information available for control, (ii) the importance to manage global influences of the local interactions, and indubitably (iii) the combinatorial nature of control possibilities with many hybrid system inputs. The research presented in this thesis aims at developing control strategies for the coordination of urban traffic networks and automated guided vehicles. We propose an approach where local controllers (also called agents) communicate with their neighbours and we devise leader-follower - type algorithms that achieve coordination already at this local or low in hierarchy level.
In a first approach, we consider the control of traffic lights in a fixed network for urban traffic, where agents at some intersections are leaders that require neighbouring intersections to coordinate their actions with given leader decisions. As a second approach, the coordination of automated guided vehicles is achieved by locally applying a set of priority rules, which can be seen as determining the leader in case of conflict. Both approaches should minimise the necessity of supervising the network at a centralised level. This can be crucially important for scalability to very large networks. Indeed, an important gain of the decentralisation is the significantly reduced communication between agents at the local level and with an eventual centralised level. Combining purely decentralised control decisions with coordinative elements, we obtain cooperation between the agents while using partial information from their neighbours only. We show on concrete case studies how this can significantly improve the performance of traffic applications. The thesis contributions include the solutions' conception,

development and implementation (in Matlab, Java), and are summarised as follows:

• *For urban traffic networks, a new platoon based model of traffic behaviour is developed and presented in Chapter 3. By grouping the vehicles into platoons, the resulting model provides an abstract representation of the dynamical evolution of the traffic state. The validity of the model is justified through the analysis of real measurements. The computational load of our model is then strictly determined by the number of platoons, and it does not change when the traffic load increases since this just entails larger platoons. The platoon based model is a basic tool for the real-time estimation and feedback control applications developed later in this thesis, by allowing fast discrete event simulations of possible traffic situations (past and current hypotheses for estimating traffic state; respective future states for comparing possible control actions).*

• *Solutions for the estimation of urban traffic networks are developed in Chapter 4. First, a centralised particle filter using the platoon based model is proposed, to generate real-time estimations of queue sizes and of platoon sizes and positions in small networks. This solution requires a centralised knowledge about the whole network in order to evaluate every single particle. Therefore, with a view to applications for larger networks, we propose a distributed particle filter, where incoming traffic information remains local and only weights of particles have to be communicated. It is still unavoidable that for larger networks the computational complexity grows, since the number of particles must be increased to ensure a good coverage of the increased number of possibilities; however, our distributed solution allows to significantly reduce communication cost. We evaluate both the centralised and distributed particle filter solutions with a detailed simulator of urban traffic, called SUMO. The obtained results show that the particle(s) with the highest weight(s) convey reasonable information on true system behaviour. Thus, we expect that the output of our estimators can be used for control.*

• *In Chapter 5, a new distributed feedback control framework is developed for the control of urban traffic networks under intermediate load. Local*

control agents use platoon-based model simulations to evaluate the quality of possible feedback control actions and hence to select the switching times of the traffic lights in the network in some optimal way. Each local control agent only needs to know the incoming traffic that already entered on its upstream roads. We divide the network into leader and follower intersections, where leaders only minimise their local queue sizes, and followers in addition take into account the implications of their switching choices on the timing of traffic sent to neighbouring leader agents. We show that, at least on academic examples, our approach is able to advantageously combine a real-time reaction to traffic arriving at local intersections with an emerging green wave type coordination throughout the network, where platoons of vehicles travel through consecutive intersections without having to stop. The examples confirm the superior performance of the leader-follower framework, compared to a fully distributed control strategy where the agents locally optimise without taking into account the implications of their switching choices on their neighbours. Compared with a nominal green wave, our approach can adapt to the real-time traffic variations as long as the proposed changes do not counteract the beneficial effects of a green wave. The benefits of combining both coordination and local optimisation, are particularly visible on non-Manhattan grids. Without a doubt, more experiments have to be performed to ensure the validity of the entire setup in practical situations. The promising results create the premises for improvements of the adaptive control methodologies currently in wide use.

• *Apart from urban traffic networks, we have also investigated the coordination of automated guided vehicles in open space, see Chapter 6. Each vehicle features a local control agent which must ensure that a delivery task is completed on time, while at the same time guaranteeing safe operation by automatically avoiding collisions. We set up a rule-based controller governing the movement of each local agent, such that a set of agents observing each other in a situation of potential conflict apply priority rules that lead to their coordination. The communication and computational complexity are thereby significantly reduced compared to a centralised*

supervisory control solution. We test the proposed algorithm with several scenarios meant to model a container terminal. We observe that, in a typical scenario including uncertainties in detections, our solutions allows to locally resolve about 85% of arising conflicts.

These contributions tackle the problem of distributed estimation and control for interacting hybrid systems in two general contexts: (i) a very structured system with fixed physical links; and (ii) a system whose topology is determined by the state changes of the local agents. We envision that insight gained from these examples could inspire controller designs for other applications of these types.

Overzicht

De gedistribueerde regeling van gekoppelde hybride systemen combineert verschillende uitdagingen (i) gedistribueerde terugkoppelregelingen gebruikt enkel lokale on-line data maar ook enkel lokale off-line modellen, (ii) het regelontwerp moet rekening houden met de globale gevolgen van lokale interacties en beslissingen, en uiteraard (iii) het combinatorische karakter van alle mogelijke stuuracties in aanwezigheid van een groot aantal hybride ingangen. In dit proefschrift worden de resultaten gepresenteerd van onderzoek naar het ontwerpen van regelstrategieën voor het coördineren van stadsverkeer en voor het coördineren van autonome voertuigen. In de hier voorgestelde aanpak kunnen lokale regelagenten communiceren met naburige regelagenten. Ten einde de regeling zo lokaal mogelijk te maken, met minimale tussenkomst van een centrale supervisor, stellen we een leider/volger paradigma voor waarbij coördinatie gebeurt via communicatie tussen een lokale leider-agent en hun by een naburige lokale volger-agenten.

In een eerste gedeelte bespreken we de regeling van verkeerslichten in een netwerk voor stedelijk verkeer. Regelagenten voor sommige kruispunten worden als leiders geselecteerd en sturen informatie naar naburige volger-agenten zodat die hun acties coördineren met de geplande acties van de leider-agent. In een tweede gedeelte bereiken we de coördinatie van autonome voertuigen door lokaal prioriteitsregels toe te passen, die kunnen worden gezien als een leider-bepaling in geval van conflicten tussen individuele trajecten. De bedoeling voor de twee situaties is de vereiste tussenkomsten van een superviserende regelaar tot een minimum te beperken. Dit leidt tot een coördinerende terugkoppelende regeling die schaalbaar

is naar zeer grote netwerken. De voorgestelde aanpak beperkt de vereiste communicatie tussen lokale agenten en elke regelagent moet enkel het lokale model kennen. De combinatie van puur gedecentraliseerde regeling en coördinerende acties zorgt ervoor, dat coöperatie tussen agenten opgezet kan worden terwijl alleen lokaal informatie wordt uitgewisseld. Verschillende voorbeelden illustreren hoe dit de prestaties van verkeersnetwerken kan verbeteren. Dit proefschrift bespreekt zowel het ontwerp van de regelalgoritmes alsook de implementatie ervan in Matlab en Java. De volgende onderwerpen komen hierbij aan bod:

• *Een nieuw model voor het dynamisch gedrag van stadsverkeer, gebaseerd op het groeperen van wagens in pelotons, werd tijdens dit onderzoek ontwikkeld, en wordt in Hoofdstuk 3 beschreven. Het model dat werd ontwikkeld leidt tot een efficiënte en abstracte voorstelling van de evolutie van de toestand van het netwerk. Het model werd gevalideerd aan de hand van metingen op een groot en divers netwerk van verkeersknooppunten. Het model laat toe om snelle in simulatieprogramma's met discrete gebeurtenissen te ontwikkelen voor het genereren van trajectoriën van het systeemgedrag. De vereiste rekentijd is grotendeels onafhankelijk van de verkeersintensiteit, vermits een toename van de verkeersintensiteit overeenstemt met grotere pelotons, maar niet met een groter aantal pelotons. De gegenereerde trajectoriën kunnen gebruikt worden voor toestandsschatting (door het vergelijken van gegenereerde trajectoriën uit het verleden met meetresultaten) en voor het selecteren van goede regelingangen (door het vergelijken van de effecten van verschillende regelbeslissingen op de gesimuleerde toekomstige trajectoriën).*

• *De significante meetruis op de verkeersmetingen maakt het noodzakelijk om een recursieve toestandsschatter te ontwikkelen, zie Hoofdstuk 4. In eerste instantie werd een gecentraliseerde partikel filter algoritme ontworpen voor het schatten in ware tijd van de grootte van de wachtrijen bij elk kruispunt en van de grootte en lokatie van pelotons. Deze gecentraliseerde oplossing moet, voor elk partikel van de filter, in één enkel simulatieprogramma de gegevens over alle pelotons in het netwerk vergelijken met de verkeersmetingen. Om de schaalbaarheid van de voorgestelde methode*

te garanderen wordt een nieuw gedistribueerd partikel filter ontwikkeld, waarbij modelinformatie en metingen alleen lokaal dienen gekend te zijn, en waarbij de lokale schatters enkel de gewichten van de corresponderende partikels moeten uitwisselen. Daardoor blijft de vereiste communicatie tussen lokale schatters beperkt. De kwaliteit van zowel de gecentraliseerde als de gedistribueerde recursieve schatters werd gevalideerd met behulp van data bekomen via een SUMO simulator, een gevalideerd microscopisch simulatiepakket. De resultaten tonen aan dat de partikels met hoog gewicht op elk ogenblik een goed idee geven van de ware netwerktoestand, en dus gebruikt kunnen worden voor toestand-gebaseerde regeling.

• *In Hoofdstuk 5 wordt een nieuw model-gebaseerd paradigma voor gedistribueerde terugkoppelende regeling van de schakeltijden van verkeerslichten ontworpen, dat vooral voor het geval van middelzware verkeersbelasting nuttig is. Lokale regelagenten gebruiken de gekende (of geschatte) toestand van het verkeer op het huidige tijdstip om, aan de hand van trajectoriëen bekomen door simuleren van het peloton-gebaseerde verkeersmodel, te beslissen welke schakeltijden van de lokale verkeerslichten de gemiddelde wachttijd van alle wagens minimaliseren. Elke lokale regelagent dient alleen het lokale verkeer te meten dat op zijn kruispunt toekomt. Een aantal drukke kruispunten worden geselecteerd als leiders, en de regelagent van deze leiders minimaliseert de lokale wachttijden. Deze leiders sturen ook berichten naar de regelagenten van naburige stroomopwaartse volgers, zodanig dat die stroomopwaartse regelagenten hun verkeerslichten schakelen zodanig dat pelotons wagens bij de leider aankomen op tijdstippen die het plan van de leider zo goed mogelijk volgen, d.w.z. de leider toelaten zijn capaciteit optimaal te benutten. Via simulaties op eenvoudige, academische voorbeelden van een verkeersnetwerk wordt aangetoond dat dit regelparadigma er inderdaad in slaagt om het netwerk te synchroniseren tot een "groene golf"-achtig gedrag (waarbij pelotons slechts zelden moeten stoppen), en tegelijkertijd ware-tijd aanpassingen door terugkoppelende regeling toelaat. De voorbeelden bevestigen de betere prestatie van deze leider/volger oplossing ten opzichte van een puur gedistribueerde terugkoppelende regelstrategie, waar lokale agenten*

hun wachtrijen optimaliseren zonder rekening te houden met de gevolgen van hun beslissingen op hun buren. Vergeleken met een vaste "groene golf" verbetert de leider/volger-oplossing de prestatie door coördinatie te combineren met een terugkoppelregeling. De voordelen van deze gecombineerde aanpak zijn vooral zichtbaar op netwerken met een structuur die afwijkt van Manhattan-grids. Een grondigere experimentele evaluatie zou deze positieve resultaten voor de praktijk nog moeten bevestigen. De veelbelovende resultaten laten ons hopen dat de voorgestelde aanpak nuttig zou kunnen zijn voor de verbetering van adaptieve regelmethoden voor andere toepassingen.

• *Naast stadsverkeer werd ook onderzoek verricht naar de coördinatie van autonome voertuigen die rijden in een open ruimte, zie Hoofdstuk 6. Elk autonoom voertuig beschikt over een lokale regelagent die ervoor moet zorgen dat het voertuig zijn taak tijdig afwerkt, en tegelijkertijd ook de veiligheid moet garanderen door botsingen te vermijden, gebruik makend van ruisgevoelige omgevingssensoren en beperkte communicatiemogelijkheden. Een regelalgoritme berekent de prioriteiten tussen voertuigen in conflictsituaties, en past op basis van die prioriteiten hun richting en snelheid van beweging aan. Er wordt aangetoond dat zowel de vereiste communicatiecapaciteit als de vereiste rekencapaciteit aanzienlijk kleiner zijn voor dit gedistribueerde algoritme dan voor een gecentraliseerde oplossing. De prestatie van het voorgestelde coördinerende algoritme wordt getoetst aan de hand van een typisch voorbeeld van autonome voertuigen voor het vervoeren van containers in een containerterminal. Er wordt aangetoond dat, ond anks onnauwkeurige sensoren op elk voertuig en zelfs zonder communicatie tussen voertuigen, 85% van de conflictsituaties lokaal opgelost worden zonder interventie van een superviserende regelaar.*

Samenvattend kunnen we zeggen dat dit proefschrift oplossingen aanbiedt om schaalbare algoritmes te ontwerpen voor gedistribueerde toestandsschatting (partikel filter) en coördinerende regeling op twee voorbeelden van interagerende hybride systemen: (i) een verkeersnetwerk met verkeer langs vaste fysische routes gecontroleerd met behulp van verkeerslichten; en (ii) een systeem waarbij de interactie tussen componenten voortdurend

verandert met de evolutie van de systeemtoestand. De inzichten die werden verworven voor deze twee zeer diverse voorbeelden kunnen bijdragen tot het ontwikkelen van leider/volger regelaars voor andere netwerken van interagerende systeemcomponenten.

Chapter 1

Introduction

1.1 Overview of interacting hybrid systems

The concept of interacting hybrid systems has a very broad interpretation, that may depend among others on how the systems interact with each other. First, a hybrid system exhibits both continuous and discrete dynamic behaviour. The interactions between these hybrid systems make the problem of estimation and control challenging for research. Such interactions can appear through a physical connection (e.g. roads carrying traffic between intersections forming an urban traffic network) or through mutual detection and reaction of systems at discrete instants (i.e. the coordination of autonomous vehicles moving in a free area and detecting each other's position). We call interactions through mutual detections as type_1, and interactions through a physical flow as type_2. Interacting hybrid systems similar to the ones mentioned before include logistics networks, manufacturing networks, irrigation channels, and flood control systems.

For example, within a factory, manufacturing networks consist of machines (equipment performing operations on the input units) and buffers (storage locations for the units produced by the machines), interconnected in any configuration. A logistic network consists of suppliers, factories manufacturing products, distribution centres, stores and customers. In the case of manufacturing and logistics networks, at the level where machines react to each other it is through mutual detection - type_1, while at the lower level, where they react to units, it is through physical flow - type_2. In the case of these

two examples, the continuous dynamics represent the position of any object moving in the network, whereas the discrete dynamics represent the moment when a machine takes up/releases an object. Irrigation channels and flood control are also two examples of interacting hybrid systems, as the dams and flood barriers open or close to control the water flow in connecting channels. Indeed, the flow of water is the continuous behaviour, while the opening and closing of the barriers can be modelled as a discrete switch. In the case of irrigation channels, interactions appear almost exclusively through physical flow - type_2, except at the level where different systems communicate to coordinate their actions (if this is the case).

All the systems enumerated above have as common characteristic the *directed* nature of interactions between the local systems. For type_1 interactions, there is a preferred direction of flow along physical connections between systems (e.g. a supply chain works in one definite direction, and traffic on a lane is restricted to one forward direction). Even type_2 interactions are often directed, due to noisy asynchronous detections (i.e. A detects B, but B does not detect A at the same time). Using directed interactions, we later define upstream and downstream relative position control strategies or priorities in the case of autonomous vehicles.

In this thesis we propose a new feedback distributed control framework for interacting hybrid systems (e.g. applied, but not limited to urban traffic networks) and a distributed collision avoidance algorithm for the efficient automatic control of the autonomous vehicles. The research presented in this thesis was possible thanks to the Control for Coordination of Distributed Systems (CON4COORD ICT-2007.3.7.(c)) - EU FP7 project. The aims of the project were to develop full control solutions for distributed systems in five case studies (i.e. underwater vehicles, aerial vehicles, traffic networks, automated guided vehicles, and complex machines), including control algorithms, communication, and informatics aspects.

1.2 Objective and contributions

The aim of the research presented in this thesis is the distributed estimation and control of interacting hybrid systems only using the local model and local information. The results presented in this thesis cover the CON4COORD case studies of traffic networks and automated guided vehicles. More precisely, we consider two case studies, namely

control of traffic lights in urban traffic networks and efficient collision avoidance for autonomous vehicles in a container terminal.

We start from a specific question:

"How to control a number of interacting hybrid systems, for example the traffic lights of intersections from an urban traffic network, in order to exploit their capabilities in a coordinated way, when each hybrid system makes control decisions based on locally available information only?"

The practical motivation to address the control of urban traffic networks is traffic congestion, that has developed as a major issue and concerns people more and more. This is mainly due to a fast increase in terms of demand and transportation means, associated with a slower development of the infrastructure. In [63], we found the following paragraph: *"To develop a 'general theory' for the stochastic behaviour of a traffic system is out of the question. Even if it were possible, such theory would be so complex as to be of no practical value"*. This major drawback of such a general approach motivated us to look at certain case studies and particular solutions (e.g. urban traffic network) instead of trying to find general solutions and theories. Therefore, the distributed feedback control framework that we propose in this thesis is meant to work for the case of intermediate load i.e. queues of vehicles are not always fully depleted at the end of each green period, but can be kept small at any time through smart control strategies.

With this objective in mind, we investigated the literature with the focus on control of urban traffic networks. We found different adaptive control methodologies, widely used that have some fundamental similarity to our work. Solutions with a relative acceptance in the field and real-world implementation are OPAC-[31], SCOOT-[37] , SCATS-[49], RHODES-[61]. Such systems are currently installed in various cities on different continents (e.g. SCOOT is used in cities such as London, Beijing, Toronto, etc.; SCATS is deployed in Melbourne, Hong Kong, Shanghai, Dublin, etc.). From the control architectures perspective, these approaches cover a broad range. SCOOT, as a fully centralized approach, uses a central optimizer to compute the solution for the entire network. The hierarchical approaches (i.e. OPAC, RHODES, or SCATS) have the centralization performed by a higher layer and the local controllers, installed in each intersection locally optimize the coarse decisions made by the upper layer.

The above mentioned solutions use various control variables (e.g. cycle length,

green split and offset) at different control levels to determine when the current green phase must be ended, meaning the next switching time. Our solution has as control variable only the switching time. We consider that in each intersection there is a model based feedback controller (also called agent). In order to achieve coordination between agents, they are assigned either as leaders or followers based on the "heaviest traffic load" criterion. The leader agents (controlling the most heavily loaded intersections of the network) only optimize their local queue sizes and send their switching schedules to their assigned followers. The follower agents take into account the implications of their switching choices on the timing of traffic sent to neighbouring leader agents in addition to optimizing their local queue sizes.

The goal is to simultaneously minimize the waiting time over all queues by combining a real-time reaction to traffic arriving at local intersections with a green wave type objective, where platoons of vehicles would travel through consecutive intersections without having to stop. Using the proposed framework, we reduce the necessity of supervising the network at a centralized level, which can be crucially important for scalability to very large networks.

The raw traffic measurements are too noisy to be used solely as queue size and traffic flow information for feedback control. We therefore propose a recursive filter to estimate the traffic state by combining real-time measurements with a reduced model of expected traffic behaviour. The latter is based on platoons rather than individual vehicles, in order to achieve faster implementations. This platoon based model is used for urban traffic estimation using the particle filtering framework. As it becomes infeasible to let a truly large traffic network be managed by one central computer, with which all the local units would have to communicate, we also propose a distributed version of the particle filter, where traffic estimations are obtained by local agents. Although traffic information only has to be exchanged between neighbouring agents, the distributed implementation requires the weights broadcasted overall network. We assess the quality of the platoon-based particle filters, both centralized and distributed, by comparing their queue-size estimates with the true queue-sizes in simulated data.

As already mentioned, the state representation of the urban traffic behaviour needs to be sufficiently simple and abstracted to allow fast discrete event simulations. This kind of simulation enables the development of model based state estimation and control tools. We propose a model that provides an abstract representation of the dynamical evolution

of the traffic state by grouping vehicles that travel closely together at approximately the same speed into a platoon. Note that we refer to platoons that naturally appear in urban traffic, in contrast to proposals for automated highway systems (e.g. PATH [57], SARTRE [12]) that artificially create vehicle platooning (i.e. driverless cars organise themselves into platoons). Using the platoon abstraction leads to efficient discrete event simulation tools, much faster than the microscopic models representing individual vehicles, while explicitly representing the heterogeneity characterizing urban traffic, something that is not possible with a macroscopic model. The model is validated by analysing real measurements and using synthetic traffic data from a well established micro-simulator called SUMO (Simulation for Urban MObility) [3]. We will show that the computational load of the platoon based model is strictly determined by the number of platoons, and is thus independent of the traffic load (i.e. an increased load leads to larger platoons, while the number of platoons remains unchanged). The platoon based model is a basic tool for the real-time estimation and feedback control applications developed later in this thesis, by allowing fast discrete event simulations of possible traffic situations (past and current hypotheses for estimating traffic state; respective future states for comparing possible control actions).

For the case of autonomous vehicles, we propose a distributed control solution for the movement and collision avoidance maneuvers in the free area (between the quay and the container rows) of a container terminal. The problem was introduced by Jan Tijmen Udding as a case study for the CON4COORD - EU FP7 project. Using a rule-based controller for each local agent (installed on each autonomous vehicle), a set of agents observing each other apply priority rules that lead to their coordination. The aim of the proposed solution is to enforce the safeness of operations as well as task completion in an automatic way. The communication and computational complexity are thereby significantly reduced compared to a centralized supervisory control solution. We test the proposed algorithm with several scenarios meant to model a container terminal.

To summarize, the main contributions of the research presented in this thesis are:

- a platoon based model as an abstracted state representation of urban traffic networks (with the computational load independent of traffic load),
- centralized and distributed solutions using particle filtering for the estimation of the state of urban traffic networks,

- a new distributed feedback control framework using the leader-follower concept for the coordinated control of urban traffic networks under intermediate load (i.e. the conjugated efforts of the local controllers are enhanced with the aim to combine real-time control decisions with an emerging green wave type coordination throughout the network),

- a distributed control solution using a rule-based controller for the task completion and collision avoidance of a swarm of autonomous vehicles.

1.3 Thesis structure

The thesis is organized as follows:

▶ Chapter 2 places our research in the general framework of interacting hybrid systems. Modelling networks, formed with this kind of interacting hybrid systems, represents the first step in developing new distributed algorithms for estimation and control. The purpose of the case studies here is to get insight that can hopefully be useful for other similar systems.

▶ Chapter 3 introduces a new model for urban traffic networks - the platoon based model. First, general notions about traffic modelling are presented, followed by a classification of the different models used for urban traffic in the literature. We continue with the results of the analysis of raw measurements which naturally introduce platoons as a new modelling abstraction of the urban traffic. Aspects related to the implementation and the performance of our model are detailed as well.

▶ In Chapter 4, we propose two particle filter algorithms for the estimation of urban traffic networks using the platoon based model. We start the chapter by presenting general issues related to estimation, and, in particular, the estimation of real-time traffic in urban traffic networks. Before introducing the standard particle filter concepts and the developed algorithm, we motivate why a particle filter is a sensible approach to estimation in this setting. First, a centralized particle filter using the platoon based model is proposed to generate real-time estimations of queue sizes and of platoon sizes and locations in small networks. This approach requires a centralized knowledge about the whole network in order to evaluate every single particle. Large scale urban traffic networks call for distributed estimation to keep the communication load within reasonable

bounds. Therefore, we propose a distributed particle filter approach. First, theoretical aspects are introduced, followed by the distributed particle filter algorithm. Validation results are presented using synthetic data produced by a third party microsimulator called SUMO [3].

▶ Chapter 5 introduces a new distributed feedback control framework for the control of urban traffic networks under intermediate load for two-way traffic. Using a model based feedback framework and local state information (platoons approaching the intersection), local controllers (agents) select the switching times of traffic lights of the network. Each agent only uses a local model of the controlled intersection and the links carrying traffic towards it. The coordination between the agents is achieved by using the leader-follower paradigm, as follower agents take into account the implications of their switching choices on the timing of traffic sent to neighbouring leader agents. Section 5.1 places our framework in a wider setting by comparing it with the state-of-art adaptive control methodologies currently in use. A preliminary control analysis for a single intersection under saturated, intermediate and very light traffic loads is presented in Section 5.2. Our novel approach, distributed feedback control with leader-follower coordination for urban traffic networks, is presented in Section 5.3. We include different simulation results that verify and validate our framework. The aim is to show that by enhancing the coordination between agents, better performance can be obtained compared to a fully distributed solution (i.e. agents locally optimize without taking into account the implications of their switching choices on their the neighbours) and to a fixed nominal green wave solution (i.e. the followers adopt a fixed optimized pattern of switches based on the actual traffic observed by their leader).

▶ Chapter 6 presents the problem of automatization of a container terminal in which unmanned autonomous vehicles load and unload berthed vessels. What makes the problem challenging are the asynchronous interactions between the AVs, the uncertainty present in the detection, and in communication. We develop a distributed collision avoidance algorithm using local rule-based controllers for each local agent (installed on each autonomous vehicle). The control algorithm is able to cope with the asynchronous interactions, the uncertain information coming from the sensors of the AV and with an unreliable communication channel between AVs. Validation results are presented using different scenarios.

▶ After a brief summary of each chapter and a review of the original contributions,

some general conclusions and directions for future work are presented in Chapter 7.

Figure 1.1 graphically presents the structure of the thesis and gives the opportunity to potential readers interested in a particular topic to orient rapidly. We recommend the reader to quickly go trough Chapter 2 before perusing any of the topics treated in the thesis. The chapter gives a minimal summary of the theoretical background used in the thesis and directs to more exhaustive references. The reader interested in urban traffic modelling, estimation or control can select one of the three chapters, from Chapter 3 to Chapter 5. The reader interested in the collision avoidance algorithms can directly go to Chapter 6 after reading Chapter 2.

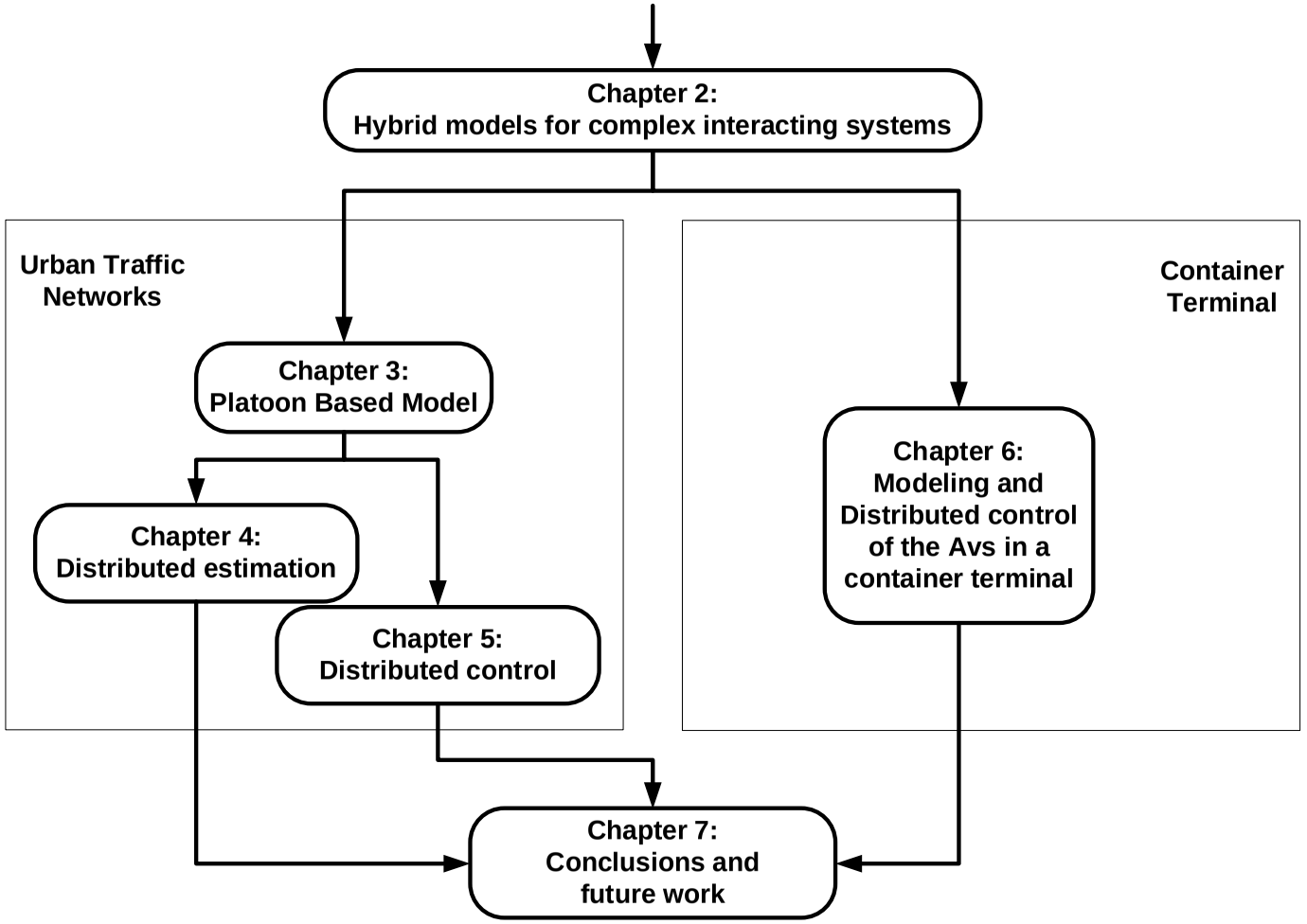


Figure 1.1: The structure of the thesis

Chapter 2

Hybrid models for complex interacting systems

In the current chapter, we place our research in the more general framework of complex interacting systems. Such complex systems consist of hybrid system components, that exhibit both continuous and discrete dynamic behaviours, interacting with each other. We start by introducing discrete event dynamic systems in Section 2.1. We also outline different ways of modelling this kind of systems like automata, timed automata and timed input/output automata. Hybrid systems are briefly reviewed in Section 2.2 together with hybrid automata, and an extension of timed input/output automata called world automata. Section 2.3 gives insight into the discrete-event simulations.

2.1 Discrete event dynamic systems

The interested reader can find a complete description of discrete event dynamic systems in [10] and [45]. Before introducing discrete event dynamic systems, we first motivate the need for a system representation.

In this thesis, we consider physical systems that consist of components acting together to perform a certain "function" that is not possible to be performed with any of the individual components. Our interest lies in developing techniques for design, control and the explicit performance analysis of a system based on well-defined criteria. Therefore, it is necessary to develop some mathematical means to describe the behaviour

of the system.

For any system we can define a set of measurable variables associated with it. From this set of variables, measured over an interval $[t_0, t_f]$, we select a subset that we can control, called input variables

$$u(t) = \{u_1(t), \ldots, u_o(t) \mid (\forall)t,\ t_0 \leq t \leq t_f\},$$

and another set, assumed to be directly measurable while varying $u_1(t), \ldots, u_o(t)$, called output variables

$$y(t) = \{y_1(t), \ldots, y_o(t) \mid (\forall)t,\ t_0 \leq t \leq t_f\}.$$

Definition 2.1. *We define $x_1(t_0), \ldots, x_n(t_0)$ as the state of the system at time t_0, the necessary information to determine at time t_0 the output $y(t)$ for all $t > t_0$ using $x(t_0)$ and $u(\tau)$, $t > \tau \geq t_0$. The set of all possible values that the state can take is called the state space X.*

The system is deterministic if no randomness is included in the evolution of its future states. Thus, such a system always produces the same output from a given starting condition or initial state. On the other hand, the output of a non-deterministic system is just defined as a set of possible outcomes. If the set of possible outcomes can be described probabilistically via distribution functions, then the system is stochastic. Next in this section the focus is on deterministic systems, but in Section 2.3, we briefly discuss the use of random-number generators in the simulation of stochastic systems.

Definition 2.2. *The model of a system is the behavioural description of that system that postulates the existence of a mathematical relationship $y(t) = g(u(t))$ between the input sequence $u(t)$ and the output sequence $y(t)$.*

A system is a real entity (e.g. the urban traffic network of a city, a plant, a vehicle etc.), while a model is an abstract representation that approximates well enough the true behaviour of that system. Further, we make a characterisation of system models based on the state.

Then, the state space model of a classical continuous-time system is

$$\begin{cases} \dot{x}\left(t\right) & = h\left(t, x\left(t\right), u\left(t\right)\right), \ x\left(t_o\right) = x_0, \ x(t) \in \mathbb{R}^n, \ t \in [t_0, \infty) \\ y\left(t\right) & = g\left(t, x\left(t\right), u\left(t\right)\right), \end{cases}$$

where $\dot{x}\left(t\right) = h\left(t, x\left(t\right), u\left(t\right)\right)$ is a set of differential equations with initial conditions specified, and $y\left(t\right) = g\left(t, x\left(t\right), u\left(t\right)\right)$ is a set of output equations.

In the case of continuous-time systems, the time t at which we define the state $x\left(t\right)$ varies continuously. For discrete-time models, time varies in a discrete way at times $T_k = k\Delta s$, whereas event-driven systems have a discrete evolution, but with the state defined only at the times when events occur. Next, we introduce an event and an event time.

Definition 2.3. *An event e_k marks a change of the state $X\left(t\right)$ in the system component(s) at a given time instant T_k (event time) i.e. if $X(T_k^-)$ is the state just prior to the occurrence of event e_k at time T_k, then the state changes to $X(T_k^+) = f(X(T_k^-), e_k)$.*

Using events, we can introduce the discrete event dynamic systems as follows.

Definition 2.4. *A discrete event dynamic system (DEDS) is a system that has a state space X and the state transition mechanism is event-driven such that to each transition from state $X(T_k^-)$ to a state $X(T_k^+)$, where $X(T_k^+)$ can be different than $X(T_k^-)$ or not, corresponds an event.*

At this point, it is important to emphasise the difference between event-driven systems and time-driven systems. In the first case, events occur at different time instants not necessarily corresponding with the clock ticks. Opposite to event-driven systems in the case of time-driven systems, the state transitions are synchronised by the clock (time) such that every clock tick corresponds to an event and only the clock advancement generates transitions (or continuous evolution for continuous-time systems).

Definition 2.5. *The set of events E, with $e_k \in E$ associated to a DEDS is called the alphabet of that DEDS, the event sequences that are allowed according to the event-driven model form "words" in that language, and a language is a set of words.*

Thus, a "language" is a formal way of describing the behaviour of a DEDS. We present the example of a queueing system, conceptually similar to the queues forming at an intersection, but much simpler.

Example 2.6. *Consider a queue formed by an infinitely large buffer and a server connected downstream at the output of the buffer. Each arriving entity either immediately proceeds to the server and is served, or waits in the queue if the server is unavailable. An entity can and will depart the buffer only after it receives service from the server.*

The DEDS system that describes our queueing example has an event set $E = \{a, d\}$*, where*

- a *denotes the arrival of an entity,*
- d *denotes the departure of an entity,*
- *and the state variable of the system is defined as the queue length and represents the number of entities waiting in the queue to be served (the queue length at time* t *the queue length is allowed to include a customer in service at time* t*). The state space of the system is the set of non-negative integers defined by* $X = \{0, 1, 2, \ldots\}$*. If* $X\left(T_k^-\right) > 0$*, then the transition corresponding to a departure event* d *at time* T_k *is* $X\left(T_k^+\right) = f\left(X\left(T_k^-\right), d\right) = X\left(T_k^-\right) - 1$*. Event* d *is not allowed to occur at time* T_k *if* $X\left(T_k^-\right) = 0$*. Also, an arrival event* a *at* T_k *is* $X\left(T_k^+\right) = f\left(X\left(T_k^-\right), a\right) = X\left(T_k^-\right) + 1$*.*

More complex examples can be thought of, e.g. several queues interconnecting servers form networks. In the case of urban traffic networks, the entities are the vehicles that travel through the network, or waiting vehicles behind the traffic light, and a server represents the ability of an intersection to allow vehicles to cross if traffic light is green. Among others, an important point is that in the case of urban traffic, the buffers have a limited capacity C. Reaching the maximum capacity leads to the blocking of the upstream intersection (server) at which the respective queue is connected (spill-back phenomenon). This implies that the event a is not allowed to occur at time T_k if $X\left(T_k^-\right) = C$.

2.1.1 Automata

We introduce here the automata framework as a way of representing languages according to some rules. The interested reader can find a complete description of automata in [41] and [10].

Definition 2.7. *A deterministic automaton G is defined as a 5-tuple*

$$G = (X,\ E,\ f,\ \Gamma,\ x_0)$$

where:

- *X is the set of finite, discrete states;*
- *E is the set of events $e_k \in E$;*
- *$f : X \times E \to X$ is the transition function ($f(x, e_k) = y$ shows the existence of an event e_k that marks the transition from state x to state y, where $x, y \in X$);*
- *$\Gamma : X \to 2^E$ is the active event function (where $\Gamma(x)$ is the set of all events e_k for which $f(x, e_k)$ is defined, and occurrence of e_k in state x is allowed);*
- *x_0 is the initial state (represented by an arrow without origin that indicates the initial state).*

The operation of automaton G starts in the initial state x_0. At the occurrence of an event $e_k \in \Gamma(x_0) \subseteq E$, G makes a transition $f(x_0, e_k) \in X$. The process continues by executing events for which f is defined in the current state one after the other.

Example 2.8. *Using the automata framework and based on the description from Example 2.6, we represent the queueing system with buffer size C through the following finite-state automaton model G:*

$$\begin{aligned} E &= \{a, d\} \\ X &= \{0, 1, 2, \ldots, C\} \\ \Gamma(x) &= \{a, d\} \text{ for all } x \in X, x > 0, \Gamma(0) = \{a\}, \Gamma(C) = \{d\} \\ f(x, a) &= x + 1 \text{ if } 0 \leq x < C \\ f(x, d) &= x - 1 \text{ if } x > 0 \end{aligned}$$

In this case, the buffer capacity is limited to C entities. The state variable x is finite and represents the number of entities in the system (in the queue and in service), and $x_0 = 0$ represents the initial state (marked by the arrow without origin). The feasible

set $\Gamma(0)$ is limited to arrival events (a) since the queue is empty, and $\Gamma(C)$ is limited to departure events (d) because the queue is full. Thus, $f(x, a)$ is not defined for $x = C$ and $f(x, d)$ is not defined for $x = 0$. Examples of words generated by the automaton G can be $ad,\ aadda,\ aaad$, etc. Figure 2.1 presents the state transition diagram of this simple queueing system.

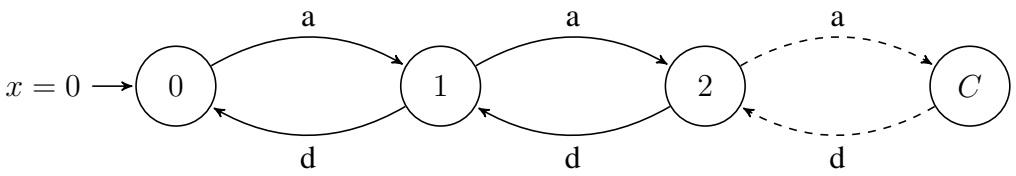


Figure 2.1: The automaton of a queueing system with limited capacity C.

For modelling a large system, the automaton can be decomposed. There are several ways of decomposing a large automaton in smaller components and of describing the interactions between these components. One way is synchronisation. That means that there are transitions with the same name in several components, and these transitions can only be executed if they are enabled in each one of the components. The following example presents a queueing system with two queues G_1, G_2, and an actuator Act, all three modelled using the synchronised automata.

Example 2.9. *The initial states are arbitrarily selected. Both queues have limited capacity C_1, and C_2, respectively. The departure input events d_1 and d_2 are enabled by Act. The actuator, controlled by a control agent CA, behaves similar to a traffic light (e.g. if G_1 has green, then G_2 has red - departure event d_1 enabled and d_2 are disabled). For simplicity, we do not allow d_1 or d_2 during the orange phases represented by R_2/O_1, and O_2/R_1. The transitions u_sw_1 and u_sw_2 can not be controlled and must occur after Δ time seconds once the states R_2/O_1 or O_2/R_1 have been reached. The time of the other two transitions c_sw_1 and c_sw_2 is selected by the control agent CA (not presented here). Figure 2.2 presents the state transition diagrams of G_1, G_2, and Act. As can be observed, G_1 and G_2 have as input events d_1 and d_2, that are output events for Act. The arrival events a_1 and a_2 can be produced as output events by upstream components "connected" as inputs to the G_1 and G_2.*

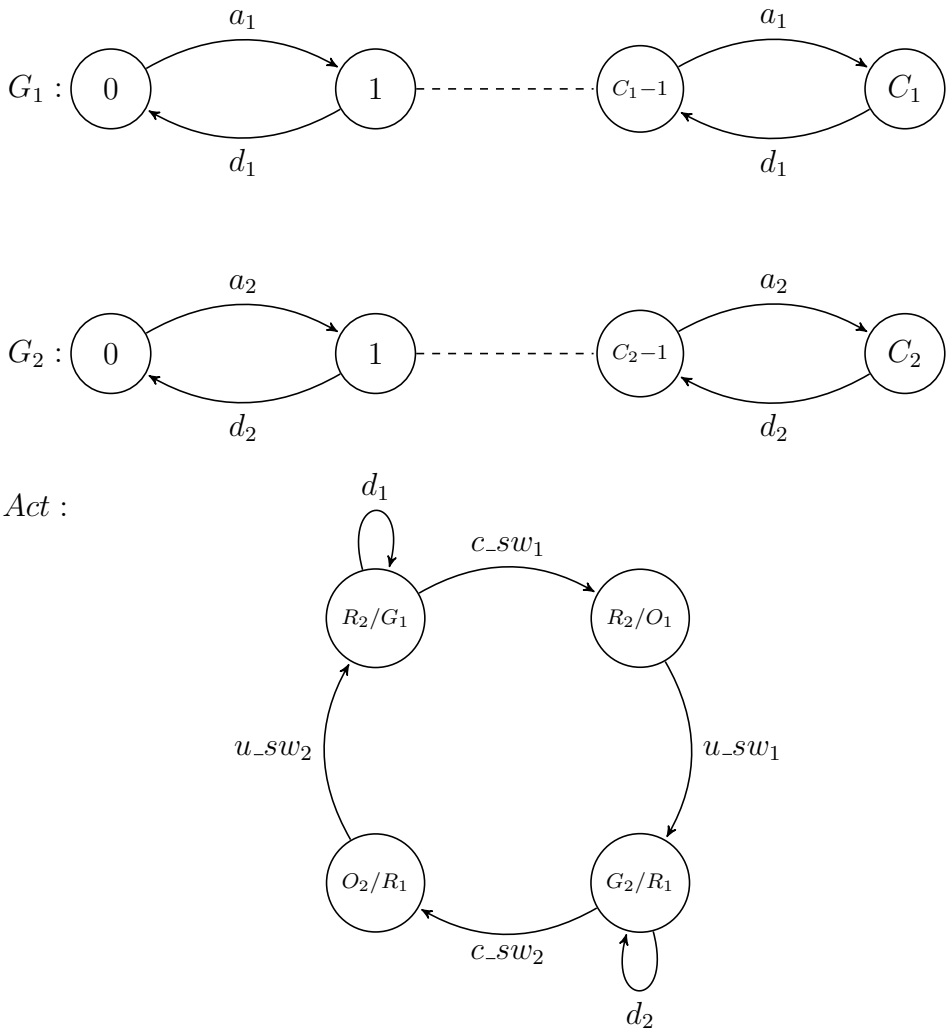


Figure 2.2: The automata of 2 competing queueing systems with limited capacities $C1$ and $C2$ and the actuator Act.

If we imagine an entire network formed by such queues and actuators, we can pose the following control problem. How could we optimise the switching schedules of all actuators from a network such that the maximum capacity from all queues is never reached?

We do not enter in details about the analysis of the DEDS modelled by automata, nor operations performed with automata. The interested reader can find more details on this

topic in [10]. A pure automaton describes the allowed sequence of events and does not give any information about the time when events occur. Next, the automata framework is extended to also include event timings as needed (e.g. to calculate performance measures in the queueing example).

2.1.2 Timed automata

In order to include time information, we need to associate to the automaton G introduced by Definition 2.7 a clock structure. The interested reader can find a complete description of timed automata in [1] and [10].

Definition 2.10. *A timed automaton G_t is defined as a 6-tuple*

$$G_t = (X, E, f, \Gamma, x_0, t_{CLK})$$

where:

- *clock time $t_{CLK} \in \mathbb{R}^+$;*
- *$X = \{x(t_{CLK}) | t_{CLK} \in \mathbb{R}^+\}$ is a set of states;*
- *$x_0 \in X$ is the initial state;*
- *$E = \{e_1, \ldots, e_N\}$ set of possible events (alphabet);*
- *$f : X \times E \to X$ the transition function inherited from automata framework;*
- *(θ_k, e_k) represents the event e_k that occurs at time θ_k;*
- *$\Gamma\left(x(t_{CLK})\right) = \{(\theta_k, e_k), k = 1, \ldots, n\}$ where $e_k \in E$ and $\theta_k \geq t_{CLK}$ (function Γ represents the list of enabled future events in state $x(t_{CLK})$ at time t_{CLK}, and is also called "event list" or " agenda" as it will be defined later on in discrete-event simulations);*

The operation of automaton G_t starts from the initial state x_0, clock $t_{CLK} = 0$, and $\Gamma(x_0)$ known, then the following sequence is repeated:

- next event [1] $e_{k^*} = \underset{(\theta_k, e_k) \in \Gamma(x)}{\arg\min} \{\theta_k\}$ is used to compute $x\left(\theta_{k^*}^+\right) = f\left(x\left(\theta_{k^*}^-\right), e_{k^*}\right)$,

[1] if multiple values yield the minimum then an arbitrary element is selected

- update $t_{CLK} = \theta_{k^*}$,
- update agenda $\Gamma\left(x\left(\theta_k^+\right)\right)$ according to specified rules (drop from the agenda (θ_{k^*}, e_{k^*}); check if new events must be added; remove events that become infeasible)

The queueing system example is already modelled as an automaton in Example 2.8. What we do next is to represent the same example using the timed automata framework.

Example 2.11. *The system described in Example 2.6 can be modelled as a timed automaton as follows:*

$$
\begin{aligned}
E &= \{a, d\} \\
X &= \{0, 1, 2, \ldots, C\} \\
\Gamma(x) &= \{(a, \theta_a), (d, \theta_d)\} \text{ for all } x \in X, x > 0, \textit{ where} \\
&\quad \theta_a, \theta_d \textit{ are different for each execution of } a, d, \textit{ respectively} \\
\Gamma(0) &= \{(a, \theta_a)\} \\
\Gamma(C) &= \{(d, \theta_d)\}
\end{aligned}
$$

$$
x\left(\theta_{k^*}^+\right) = f\left(x\left(\theta_{k^*}^-\right), e_{k^*}\right) = \begin{cases} x\left(\theta_{k^*}^-\right) + 1 & \text{if } e_{k^*} = a \text{ and } x\left(\theta_{k^*}^-\right) < C \\ x\left(\theta_{k^*}^-\right) - 1 & \text{if } e_{k^*} = d \text{ and } x\left(\theta_{k^*}^-\right) > 0, \end{cases} \tag{2.1}
$$

where the event a marks the arrival of an entity, d is an entity departure event, and C represents the maximum number of entities that can wait in the queue to be served by the server. The state $x(t_{CLK})$ represents the queue length (entity in service included) at time t_{CLK}. Since we do not have departures, if $x\left(\theta_{k^*}^-\right) = 0$, the transition function f is not defined for $x\left(\theta_{k^*}^-\right) = 0$, $e_{k^*} = d$ given that $d \notin \Gamma(0)$. Using the notations introduced before, we describe the next event e_{k^*} at time θ_{k^*} as

$$
e_{k^*} = \begin{cases} d & \textit{if } \underset{(\theta_k, e_k) \in \Gamma(x)}{\textit{arg min}} \{\theta_k\} = d \textit{ and } x\left(\theta_{k^*}^-\right) > 0 \\ a & \textit{if } \underset{(\theta_k, e_k) \in \Gamma(x)}{\textit{arg min}} \{\theta_k\} = a \textit{ and } x\left(\theta_{k^*}^-\right) < C. \end{cases} \tag{2.2}
$$

The next event e_{k^*} is determined based on the comparison of time values of the next a and d events. The exceptions are for $x\left(\theta_{k^*}^-\right) = 0$, when only arrivals are possible, and

for $x\left(\theta_{k^*}^{-}\right) = C$*, as only departures are allowed.*

We already mentioned that more complex examples of interconnected queueing systems could be thought of. Models of such systems naturally lead to high dimensions. Thus, decomposing timed system descriptions becomes important. The timed input/output automata (TIOA) modelling framework is used next to reach this desideratum.

2.1.3 Timed input/output automata

The TIOA framework is a refinement of the timed automata framework by distinguishing between input and output actions represented by events. The interested reader can find a complete description of timed input/output automata in [43] and input/output automata in [50].

Definition 2.12. *A TIOA, denoted* G_{tio}*, is a* 3*-tuple*

$$G_{tio} = (G_t, I, O)$$

where

- $G_t = (X, E, f, \Gamma, x_0, t_{CLK})$ *is a timed automaton,*
- I *the set of input events* $I \subseteq E$*,*
- O *the set of output events* $O \subseteq E$*.*

The operation of the automaton G_{tio} is similar to the execution of G_t with the only difference that some events, as input events, are externally triggered or some other events, as output events, can externally trigger transitions from the other automaton interconnected with it. The input event must always be treated by the automaton that receives it.

Example 2.13. *The following example presents a queueing system with two queues* G_1*,* G_2 *in series connection (like in Figure 2.3) modelled using the TIOA framework. Whenever* d_1 *occurs the second buffer* G_2 *must be ready to accept the resulting arrival.*

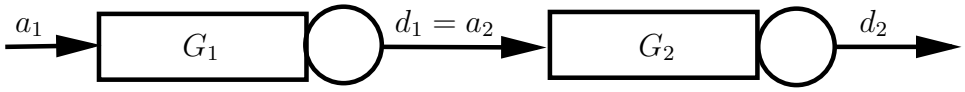


Figure 2.3: Example with two queues in series connection modelled using the TIOA framework

2.2 Hybrid systems

We have already analysed in Section 2.1 the difference between time-driven and event-driven systems. Briefly, in time-driven systems, the state continuously changes as time changes. In event-driven systems, it is only the occurrence of asynchronously generated discrete events that forces instantaneous state transitions. Our presentation is now extended to a new class of systems that include both types of dynamics.

Definition 2.14. *Hybrid systems are systems that combine time-driven with event-driven dynamics.*

The above definition is given in [10]. It is important to stress that in general a DEDS is an abstraction of a hybrid system (e.g. queues at intersections make an abstraction of the complex continuous dynamics of vehicles). Sometimes, however, a hybrid system can be seen as a simplification of a complex DEDS.

Coming back to the queueing system example, we saw in the previous section that it is a purely event-driven dynamic system. However, when the volume of entities is too large, we can use a fluid flow representation. In this approach, the arrival/departure of entities is modelled as fluid flow and the buffer is the holding tank for this fluid. The model needs to recognise when the buffer becomes empty or reaches its capacity. The representation is a hybrid system seen as an abstraction of the queueing system presented in Example 2.6. A modelling framework for hybrid systems is provided by hybrid automata, which may be seen as an extension of the timed automata, when we replace simple clock dynamics at each discrete state by arbitrarily time-driven dynamics characterizing one or more continuous state variables. Therefore, the system state is now expressed as (s, x) where $s \in S$ and $x \in X$. The set of discrete states, often called modes, is S and the set of continuous states is X. A discrete state s is usually referred to as the mode of the hybrid system (e.g. empty queue), while x often represents the

physical state of some process (in our case the number of entities - vehicles - waiting in the queue to be served).

2.2.1 Hybrid automata

A complete description of hybrid automata can be found in [39] and [10].

Definition 2.15. *A hybrid automaton, denoted by G_h, is formed by the 10-tuple*

$$G_h = (S, X, E, f, \Phi, Inv, Guard, \rho, s_0, x_0)$$

where

- $s \in S$ *is a set of discrete states or modes;*
- X *is a continuous state space (normally* $\mathbb{R}^n$*);*
- E *is a finite set of events;*
- f *is a vector field,* $f : S \times X \to X$ *where* $\dot{x} = f(x, s)$ *is the active differential equation in mode* s*;*
- Φ *is a discrete state transition mapping,* $\Phi : S \times X \times E \to S$*;*
- Inv *is a set defining an invariant condition (also called domain),* $Inv \subseteq S \times X$*;*
- $Guard$ *is a set defining a guard condition,* $Guard \subseteq S \times S \times X$*;*
- ρ *is a reset function,* $\rho : S \times S \times X \times E \to X$*;*
- $s_0 \in S$ *is an initial discrete state;*
- $x_0 \in X$ *is an initial continuous state.*

The extra added components compared with the timed automata framework are the guard, invariant, and reset conditions. A guard condition $guard$ specifies a subset of X which, when entered while the system mode is s at time T with event e, enables a transition from s to some new mode $s^*(T^+) = \Phi\left(s(T), x(T), e\right)$. A reset condition specifies how a value of x in some mode s at time T changes to a new continuous

state $x^*(T^+) = \rho(s(T), s^*(T^+), x(T), e)$ at some new mode s^* when a transition from s to s^* takes place. An invariant condition Inv (also called domain) is a subset of X associated with a mode s such that x must belong to this set in order to remain in this mode; should an invariant condition no longer hold, the system is forced to switch modes and the precise transition needs to be defined through ϕ. A general condition that has to hold for each state of the automaton is $Guard_{(s(T),s^*(T),x(T))} \subseteq Inv_{(s(T))}$.

Similar to automata framework, in the execution of a hybrid automaton, a deadlock can occur when the state cannot evolve either in a time-driven or event-driven fashion (e.g. the invariant condition ceases to hold, but no consistent guard condition is specified to force a mode transition). The Zeno behaviour appears if the state evolution of the hybrid system involves an infinite number of discrete transitions in a finite amount of time (can also occur in the case of timed automata). The solution is to explicitly include constraints to require a lower bound on the time between successive discrete transitions.

Example 2.16. *Coming back to our queueing system example, an external controller (in our application the agent controlling the traffic lights) changes from time to time the outflow rate $\mu(t)$.*

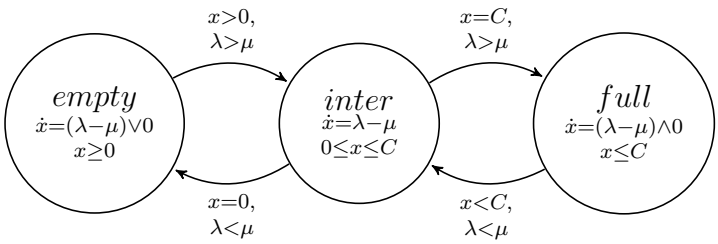


Figure 2.4: The hybrid automaton of a queueing system with limited capacity C.

The inflow rate $\lambda(t)$ is a time dependent random process that describes the arrival rate of vehicles in the queue. For simplicity we drop the time index of λ and μ for the rest of the presentation. Clearly, λ and μ are positive-semidefinite. The model captures the states when the buffer becomes empty (state "$empty$" in which $x = 0$) or when it is at full capacity (state "$full$" where $x = C$). The middle state '$inter$" represents the situation $0 < x < C$. The initial state is arbitrarily selected. The state transition diagram is presented in Figure 2.4 with the following elements:

- *state "$empty$":*

$$f\left(empty, x\right) = \left(\lambda - \mu\right) \vee 0$$

$$\Phi\left(empty, x\right) = \begin{cases} inter, & \textit{if } x > 0 \\ empty, & \textit{if } x = 0 \end{cases}$$

$$Inv_{empty} : \; x \geq 0$$

$$Guard_{(empty,\, inter,\, \lambda > \mu)} : \; x > 0$$

- *state "$inter$":*

$$f\left(inter, x\right) = \lambda - \mu$$

$$\Phi\left(inter, x\right) = \begin{cases} empty, & \textit{if } x = 0 \\ inter, & \textit{if } 0 \leq x \leq C \\ full, & \textit{if } x = C \end{cases}$$

$$Inv_{inter} : \; 0 \leq x \leq C$$

$$Guard_{(inter,\, empty,\, \lambda < \mu)} : \; x = 0$$

$$Guard_{(inter,\, full,\, \lambda > \mu)} : \; x = C$$

- *state "$full$":*

$$f\left(full, x\right) = \left(\lambda - \mu\right) \wedge 0$$

$$\Phi\left(full, x\right) = \begin{cases} inter, & \textit{if } x < C \\ full, & \textit{if } x = C \end{cases}$$

$$Inv_{full} : \; x \leq C$$

$$Guard_{(full,\, inter,\, \lambda < \mu)} : \; x < C$$

Again, we will not give here more details about hybrid systems. The extension to Hybrid Input/Output automata (HIOA) can be done similar to the case of the TIOA framework. The interested reader can find a complete description of HIOA in [51].

Next, we introduce an extension of the HIOA framework called World Automata (WA) used to model the autonomous vehicles case study presented in Chapter 6.

2.2.2 World automata

The world automata framework, introduced in [8], extends the HIOA modelling framework of [51] by specializing some variables, called *world variables*. Moreover, world variables represent the changes in the environment as they might be perceived by the different agents (autonomous vehicles in our example). The main difference between world and standard automaton variables is that world variables are a function of time and space, not only of time as in standard automaton variables. Hence, world variables values (and trajectories) will depend both on the time instant and on the position in the underlying space. The "world automata" name comes from the aim to represent the connection between the agents and the surrounding world.

Thus, automaton G_{WA} representing an agent uses its world inputs U_w to receive information from the world it lives in. Analogously, it will use its world outputs Y_w to give information to the world it lives in. Finally, internal world variables X_w are used to represent the world characteristics of G_{WA}. World variables are used by the automaton to catch environmental changes and to represent the automaton influence to the environment. The environment is represented as a HIOA too. Results are extended in [8] to represent both an *agent* living in a world and a *world* for other agents living in it. World variables will then be used for implicit communication between agents and worlds at different levels of depth. As a consequence of this representation, a mechanism to distinguish between the world variables is used.

The WA framework is used in this thesis to describe the case study presented in Chapter 6. As joint work with the Electronic Systems Design group, University of Verona, our contributions are the system architecture, the control algorithm, and the implementation and validation of the model. Therefore, the focus here is not on presenting in more details the theoretical aspects of the World Automata framework. The interested reader can find a complete description of operators on WAs in [9] and references therein.

2.3 Discrete-event simulations

So far, we have presented different ways to obtain a model which describes the behaviour of a system. We introduced the automata, timed automata and timed input/output automata frameworks used to describe the evolution of the state of a DEDS as a result of event occurrences over time. In general,"real world" systems either do not conform to some assumptions we make in order to simplify a model, or they are just too complex to yield analytical solutions. In this cases, through simulation, we must numerically evaluate a model. A complete description of discrete-event simulations can be found in [10].

For stochastic systems, the simulation may be viewed as a systematic method of generating possible sample paths of a DEDS. In order to simulate stochastic DEDS, we need random number generators used to obtain samples from various probability distributions of interest (e.g. random service duration for queueing models with given probability distribution). These samples provide the clock structure of the stochastic timed automaton model to generate a sample path. The simulation of a stochastic system, like finite state stochastic processes, allows calculating the average behaviour according to the probability of each sample path by generating a histogram, i.e. a set of equally likely sample paths. Since we have not fully introduced stochastic models, we limit our presentation here to only include a reference to the use of random-number generators.

The components of a discrete-event simulation (DES) are:

- event(s) - each event has associated a time and a type marking the change in the state of the system. The events are instantaneous;
- events list (agenda) - the time ordered list of the future events that are pending as a result of previously simulated events but have yet to be executed themselves by the simulation program. It is important for the correctness of the simulation to keep the agenda ordered;
- simulation clock t_{CLK} - the current simulation time, in the measurement units suitable for the system being modelled. The time when an event occurs triggers the simulation clock to move to the next event time (minimum time in agenda) as the simulation proceeds;

- random-number generators - depending on the system model, the simulation needs to generate random variables of various kinds;
- ending condition(s) - theoretically, a discrete-event simulation could run forever, but in practice the simulation is ended using different conditions (e.g. when $t_{CLK} >$ a given time);

At the initialisation, the agenda is populated with events and t_{CLK} gets the time of the first event in the agenda. When this first event from the agenda occurs, at time t_{CLK} , the state of the model is updated according to the type of event and the current state. The executed event is deleted and the new events that could result from the execution of the first event can be generated into the agenda, possibly using random number generators to determine their type and their future execution time. The newly generated events are introduced into the agenda. Time t_{CLK} is updated with the time of a new first event from the agenda and the process is reiterated. The simulation continues until one of the end conditions is met or until the agenda is empty.

In the case of hybrid systems, since both time-driven and event-driven dynamics are present, we have to treat them together. Therefore, the simulation replaces the continuous-time model by a Δt increment approximation to simulate in discrete time the evolution of continuous states between events.

2.4 Summary

The chapter briefly introduces the discrete event dynamic systems and hybrid systems. Modelling frameworks used to represent these kind of systems are briefly described together with the example of modelling a queuing system. The discrete-event simulations are also presented as simple ways to numerically evaluate the obtained models by generating all possible sample paths.

Chapter 3

Modelling urban traffic networks

Traffic models are used to forecast the future traffic states of a traffic network (urban or freeway). The forecast is then used for making decisions such that the network meets certain criteria (eg. minimum waiting time in front of the traffic lights or in the case of freeways the average speed is above a given value, etc.).

A new platoon based model (PBM) of traffic behaviour in an urban traffic network is proposed in this chapter. The PBM allows fast discrete event simulations, and hence enables model based state estimation presented in Chapter 4 and model based feedback control tools for urban traffic presented in Chapter 5.

Section 3.1 starts with a representation of the main components of an urban traffic network (link and intersection) followed by a classification of the traffic models in Section 3.1.3. Next, we give a motivation for using platoons as a reduced model for urban traffic flows in Section 3.3 based on real measurements presented in Section 3.2. The detailed description of the PBM is given in Section 3.4, and Section 3.4.6 represents it as an hybrid input/output automaton, using a composition rule as in [50]. Details about the implementation of the model and an assessment of the model is made in Section 3.5.

3.1 Introduction

3.1.1 Link modelling

The simplest way for describing the 1-dimensional way trajectory of a vehicle over time is to describe its position at any time t as a function $x\left(t\right) = x(0) + \int_0^t v\left(\tau\right) d\tau$ where $v\left(t\right) = \frac{dx(t)}{dt}$ is the velocity of the vehicle at time t. This representation is called microscopic model. Using this for a large number of vehicles becomes computationally infeasible. Moreover, in practice it is very hard to obtain online information from each vehicle involved in traffic.

A realistic scenario is to have information from many fixed locations and based on that to obtain an averaged interpretation of the traffic streams. An average traffic description reduces the computation burden as a trade-off for reducing the level of details. This kind of model is called macroscopic model and it only deals with average results over several red-green cycles instead of optimizing each individual cycle as we do for the rest of our work. Basic elements of the average traffic behaviour are:

- the number of vehicles $n_i([x, x + L], t)$ observed at time t between location x and $x + L$, and the number of vehicles $n_f(x, [t, t + \Delta T])$ observed at location x during time ΔT,
- the average speed $v(x, t)$ measured at location x at time t,
- the *density* $\rho\left(x, t\right) = \frac{n_i([x,x+L],t)}{L}$ at time t (often denoted as $k(t)$ in traffic theory) defined as the number of vehicles n_i per area L (e.g. road section, link, etc.),
- the *flow* $\lambda\left(x, t\right) = \frac{n_f(x,[t,t+\Delta T])}{\Delta T}$ at time t (often denoted as $q(t)$ in traffic theory) as the number of vehicles n_f passing a fixed location per unit of time (e.g. second, hour etc.).

Based on these 3 elements characterizing the average traffic behaviour, we can define the following traffic flow equation

$$\lambda(x, t) = v(x, t) \cdot \rho(x, t). \tag{3.1}$$

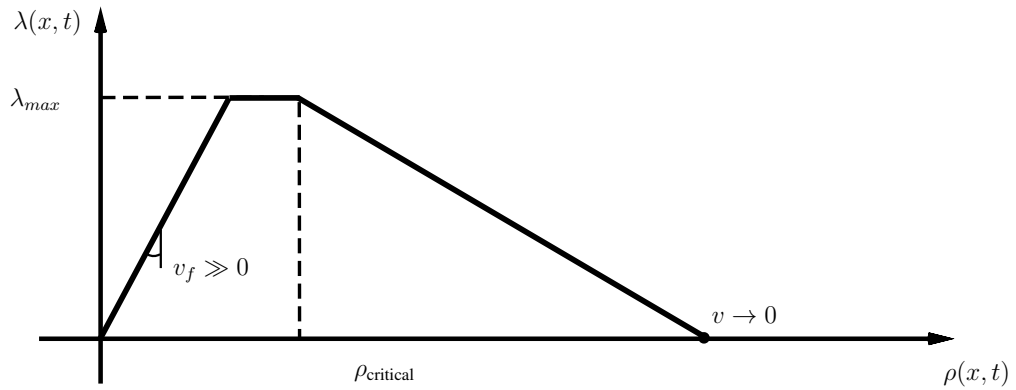


Figure 3.1: Approximation of the macroscopic fundamental diagram - v_f called free velocity and represents the maximum legal velocity on a free lane.

The approximation of *macroscopic fundamental diagram* (MFD) depicts the relation between the traffic flow $\lambda(x,t)$, traffic density $\rho(x,t)$, and average speed $v(x,t)$. Note that the term macroscopic fundamental diagram is also used for the so called network fundamental diagram as in [21, 33]. The MFD consists of 3 different two-dimensional graphs (flow-density, speed-flow, and speed-density) based on (3.1).

Figure 3.1 shows the flow-density diagram for a sufficiently long link having the maximum legal velocity v_f. The speed also varies along the curve presented in Figure 3.1, although its values are not represented. The value of the speed at one point on the curve is represented by the slope of the line connecting that point to the origin. The diagram has three branches. The first branch, starting from the origin with a positive slope, includes "very light" traffic followed by"undersaturated" phases (explained later). The second part of the graph is the intermediate branch when the maximum flow rate λ_{max} is reached. This part ends when the density reaches $\rho_{critical}$ for which the maximum number of vehicles λ_{max} can pass by a point in a given time period and it leads to instability. The third part, with a negative slope, represents the congested branch. Even though there are more vehicles on the road (than for the previous two branches), the flow rate λ is reduced compared with the number of vehicles passing the same location if there were fewer vehicles on the road. The limits, v_f the maximum legal speed and v as the speed goes to zero, represent the two extreme points of graph.

The MFD has been shown to be useful for freeway traffic control (sufficiently long

links). For urban traffic, the existence of MFD is not always proven to be true. In [21, 33], the existence of analytical approximation of MFD is proven and an experimental MFD is established for any link (urban street) with blocks of diverse widths and lengths, but no turns, even though some intersections are controlled by arbitrarily timed traffic signals. If the congestion is unevenly distributed the network exhibits scattered traffic states that do not fit in the MFD diagram. This situation is typical for large urban areas with multiple congested sub-centres or for urban networks with rapidly changing demand.

3.1.1.1 Transient behaviour

Urban traffic exhibits different phases like:

- *very light* traffic: situation occurs during the night when queues do not form at intersections or are very rare.
- *undersaturated* traffic: a traffic situation when there exists a periodic switching such that the queues formed behind the traffic lights will be fully depleted at the end of the current green period.
- *intermediate* traffic: queues are not always fully depleted after each green period, but through smart control strategies they can stay bounded. Only a few critical intersections of the network are heavily loaded.
- *saturated* traffic: during rush hours, when regardless of green switching strategies the queues are present at the end of each current green period (rarely depleted). The spill-over can appear as an extreme case (oversaturated traffic).

A spill-over happens when the number of vehicles waiting behind the traffic light from cycle-to-cycle grows until the limited capacity of the link between the intersections is reached and the up-stream intersection is blocked. An identification and analysis of the spill-overs in the urban traffic environment is presented in [34].

3.1.2 Intersection modelling

The simplest continuous time model of a generic intersection I_j is by modelling the average behaviour of the traffic. Then the arriving traffic measured in number of vehicles

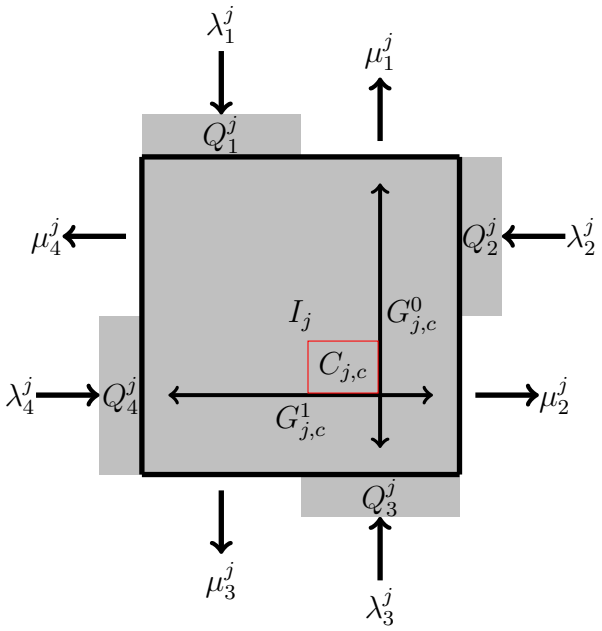


Figure 3.2: Fluid flow model of intersection I_j

per units of time (U.T.) at the entrance i of intersection j is λ_i^j (inflow) and similar the departure rate of traffic μ_i^j (outflow).

Without loss of generality, we denote once and for all the entrances of a typical intersection like the one from Figure 3.2 as follows: 1 is North, 2 is East, 3 is South, and 4 is West.

The intersection is actuated by traffic light that has the cth switching cycle

$$\text{cycle}_{j,c} = G_{j,c}^0 + G_{j,c}^1 + 2 \cdot O \tag{3.2}$$

where $G_{j,c}^{d(i)}$ represents the green period for a pair of entrances (e.g. $G_{j,c}^0$ for the 1 and 3 respectively $G_{j,c}^1$ for 2 and 4) and O is the orange period.

Remark 3.1. In our model, which will be introduced later , the acceleration and deceleration of the vehicles are neglected; also we assume that the orange period O is 0. We further call this ”stop-and-go” traffic.

Figure 3.2 depicts a typical intersection with 4 entrances and for outputs. Using the continuous model, we can define the queue of vehicles $Q_i^j(t)$ at time t at entrance i of

intersection j as follows

$$\frac{d}{dt}Q_i^j = \lambda_i^j - \mu_i^j G_{j,c}^{d(i)}(t), \tag{3.3}$$

provided that $Q_i^j(t) > 0, \forall i, t$ and $G_{j,c}^{d(i)}(t) = 1$ when green for $Q_i^j(t)$. If at any time $t,\ Q_i^j(t) = 0, \forall i$ and $\lambda_i^j < \mu_i^j$ then the queue remains zero.

The interested reader can find more aspects of traffic theory in [20].

3.1.3 Urban traffic model classification

Many papers have dealt with the modelling and control of essentially one-dimensional freeway traffic. However we consider urban traffic, where the motion of vehicles is more complicated given by the 2-way traffic, turns, stop-and-go behaviour, and by the interactions with the traffic lights. As a result, urban traffic exhibits more spatial and temporal diversity. A classification of the urban traffic models can be made based on the details level:

- microscopic models represent the dynamics of each vehicle (”car following” models as in [13]) or ”in each small cell” (as in SUMO [3], RoadSim [2], and TRANSIM [62]), leading to an excessive computational cost for real-time use with controllers. Opposite to the level of abstraction provided by the microscopic models are the macroscopic models.

- macroscopic models (e.g. TransCad [74], Emme [30], PTV Vision [69], Cube Voyager [17] or [6, 11, 24, 73]), describing traffic flow averaged in space and time over fixed intervals, allow fast computations, but cannot represent the traffic heterogeneity that must be considered in order to optimally use the capacity of signalised intersections.

- mesoscopic models (e.g. Cube Avenue [16], Dynameq [29], OmniTRANS StreamLine [66]), as a third category, are a combination between the first two model categories, microscopic models and macroscopic models.

A large class of urban traffic models are the so called the spatio-temporal discrete models, where the space (links connecting the intersections) and the time are sampled, like: a store-and-forward model is proposed in [32], a cell-transmission model in [18, 19] or a link-transmission-model in [82].

3.2 Real measurements available for the traffic analysis

The available measurements are taken over 40 days in the area of Dendermonde, Belgium as shown in Figure 3.3 (data courtesy of the "Vlaamse overheid" - Belgium).

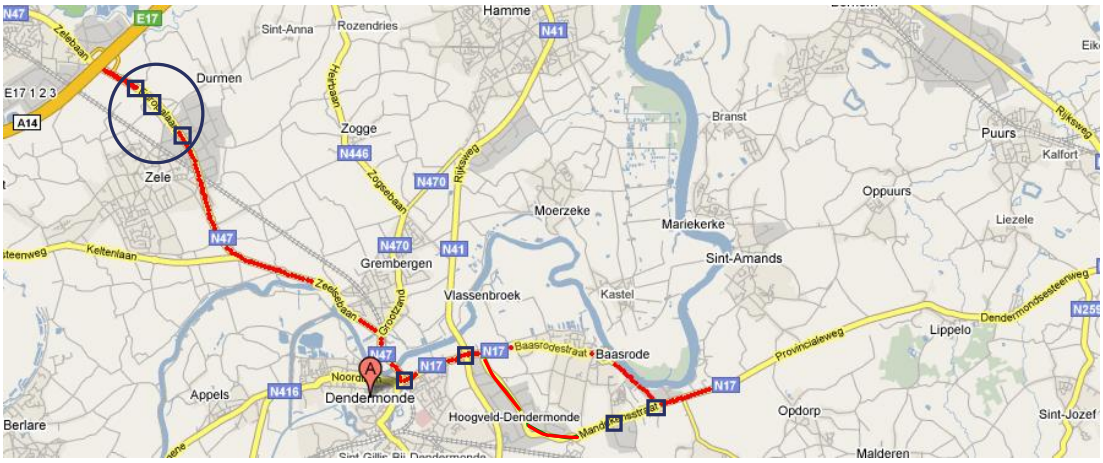


Figure 3.3: Area of Dendermonde between E17 - N17 where the measurements are taken (network indicated by the red line, signalised intersections indicated by rectangles, and area on which we focused encircled).

The covered area contains a traffic network, like in Figure 3.4, with 7 signalised intersections (indicated by rectangles), several unsignalised intersections and 4 roundabouts. In our study, we focused only on the North-Western part of the network (indicated by a circle in Figure 3.3) that contains 3 signalised intersections.

The traffic network includes sources of oversaturated traffic during rush hour (e.g. Dendermonde city, train station - very non-stationary behaviour, and the industrial area) combined with sources of uncertainty (urban area and hospital). Measurement locations are selected just before and after the intersections and roundabouts, perpendicular to the traffic flow direction. Additional sensors between the intersections — embracing more than 100 sensor locations in total — increase the quantity (and thus combined accuracy) of measurements significantly above the typical traffic estimation context.

Pneumatic road tube sensors are used to obtain the measurements. Such sensors send a burst of air pressure along a rubber tube when a vehicle's tires (axle of a vehicle) pass over the tube at the sensor location. The pressure pulse closes an air switch,

producing an electrical signal that is transmitted to a computer that logs the information.

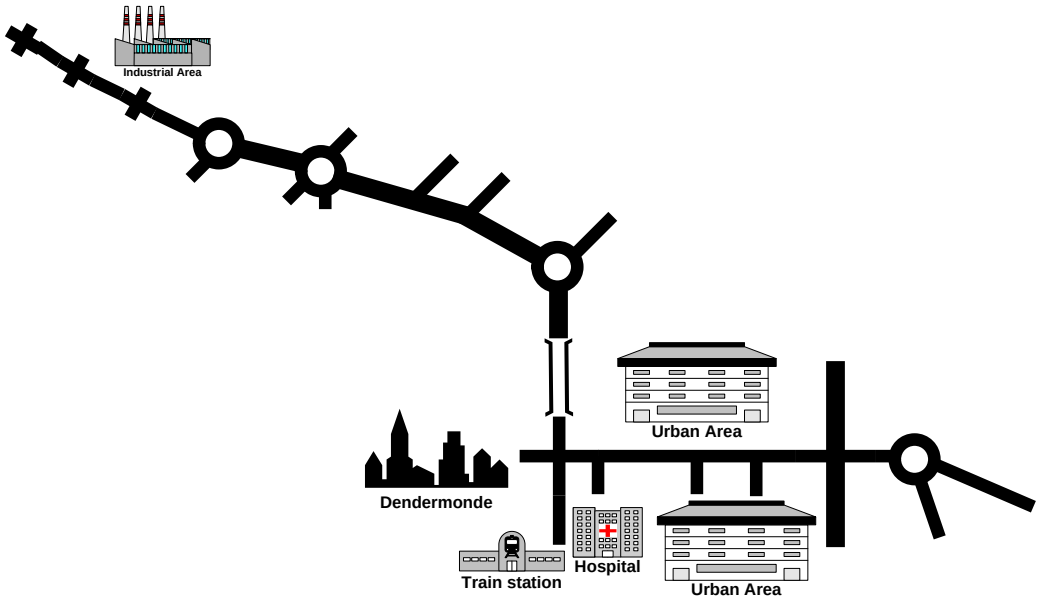


Figure 3.4: The traffic network in Dendermonde area.

Despite the quick installation and low cost, this type of sensors suffers from the inaccurate axle counting when traffic volume is high, temperature sensitivity of the air switch, and low life expectancy, leading to many errors, both missed detections and false detection. The obtained measurements do not allow to distinguish whether a large time delay between two successive pulses is due to a large distance between vehicles or to a very low speed of close vehicles. In the considered context however, the consistent majority of vehicles either have a speed just below the maximum legal speed limit or are stopped (e.g. at a red light). All these reasons lead to a high number of errors, so we need a good approximation for a further analysis of the measurements.

3.3 Platoons in urban traffic

We want to show that in urban traffic, groups of vehicles typically travel closely together at approximately the same speed, behaving as one coherent platoon, due to the overtaking limitations [12], [15], and as a result of the interaction with the red/green

cycle of traffic lights. In our case, the platoon based model, is an abstract dynamic stochastic model of urban traffic which is sufficiently simple and computationally fast to allow real-time comparison of several possible alternative estimates presented in Chapter 4 (and, later, control actions presented in Chapter 5).

By grouping the vehicles into platoons, the resulting model provides an abstract representation of the dynamical evolution of the traffic state. We expect that this leads to efficient discrete event simulation tools, much faster than the microscopic models representing individual vehicles, while explicitly representing the heterogeneity characterizing urban traffic, something that is not possible with a macroscopic model and that is needed for the efficient control of traffic lights. The validity of the platoon based model is justified through the analysis of real measurements.

3.3.1 Experimental validation of platoon based vehicle aggregation

To investigate how accurate such a platoon model might be, we analysed the available measurements from the Dendermonde traffic network.

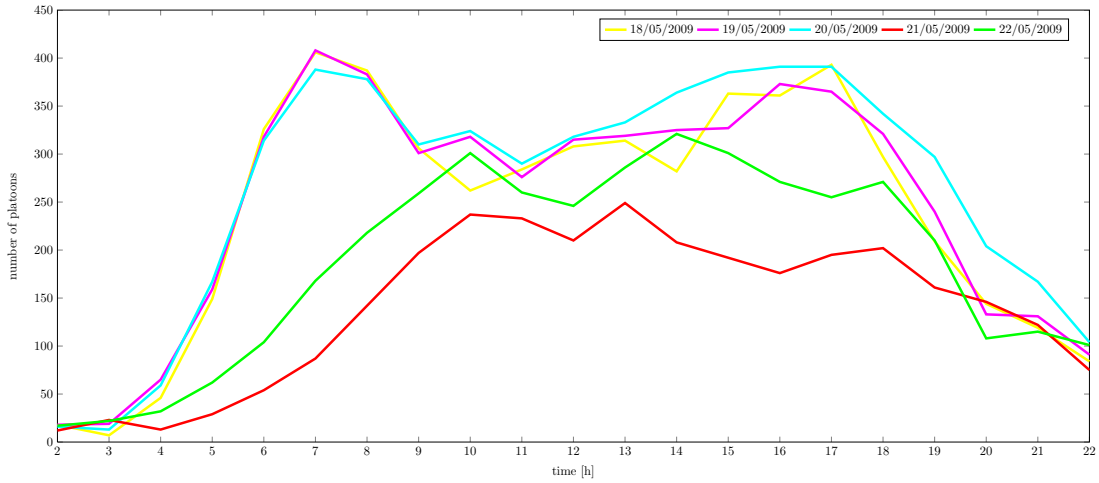


Figure 3.5: Number of platoons (defined with maximal inter-vehicle time $\Delta = 5[\text{sec}]$) during the working days of 1 week.

We therefore attempt to aggregate vehicles in platoons according to the following

rule: consecutive sensor pulses belong to the same platoon if they are separated by less than Δ seconds. For Δ between 3 and $10[\text{sec}]$, we found that most vehicles travel in platoons of more than one vehicle, thus indeed leading to model reduction (except in very light traffic, e.g. during the night, when a model for traffic control is not needed anyway). The number of platoons does not significantly decrease when increasing Δ beyond $5[\text{sec}]$, so we select $\Delta = 5[\text{sec}]$ in further analysis.

As a conclusion, on the basis of available data, we intend to define platoons in the context of urban traffic networks with intermediate traffic load as follows: either vehicles move close to nominal speed, and platoons are robustly defined on the basis of $\Delta = 5[\text{sec}]$; or vehicles join/leave a queue at a traffic light, and platoons are defined by how the vehicles merge into the queue and leave the intersection together at green.

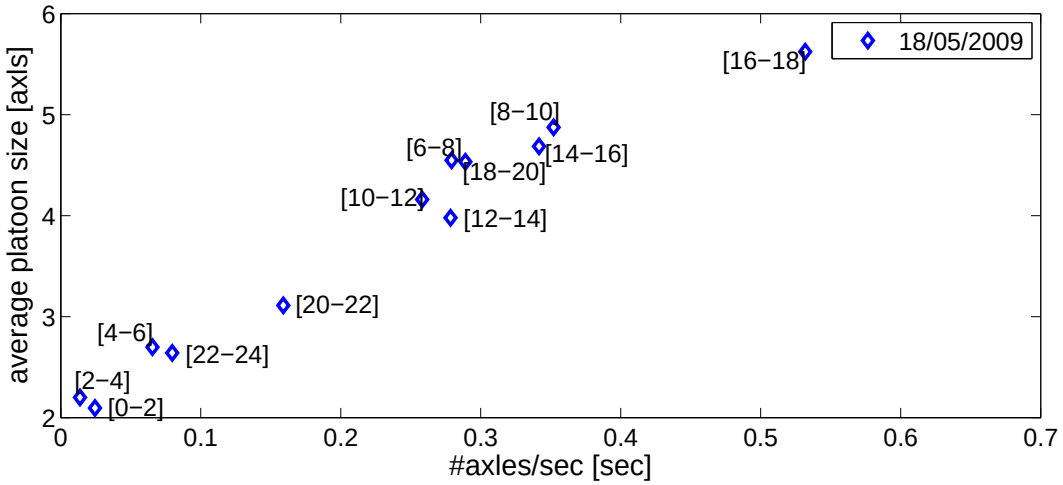


Figure 3.6: Average size of the platoons (defined with maximal inter-vehicle time $\Delta = 5[\text{sec}]$) vs. traffic intensity for one working day with 2 hours step.

Remark 3.2. A large Δ leads to a large average platoon size, the number of "single vehicle" platoons will decrease and at the limit all vehicles will belong to one big platoon. Thus, one big platoon does not provide any useful information about the location of vehicles. On the contrary, a small Δ makes a small average platoon size and no platoons can be identified. At the limit a small Δ leads to a microscopic traffic representation with a high computational cost. Therefore, choosing Δ is very important and it may have to be adjusted according with the time, weather and also the location (e.g each country/continent has its one peculiarities in terms of urban traffic).

Figure 3.5 shows that the *number* of platoons present in the network at any time does not vary by more than a factor 2 over all daytime hours of a working week, although flow rate can vary a lot. Indeed, further analysis – see Figure 3.6– confirms that flow rate rather affects the average *size* of the platoons, which grows roughly proportionally with the number of axles per second.

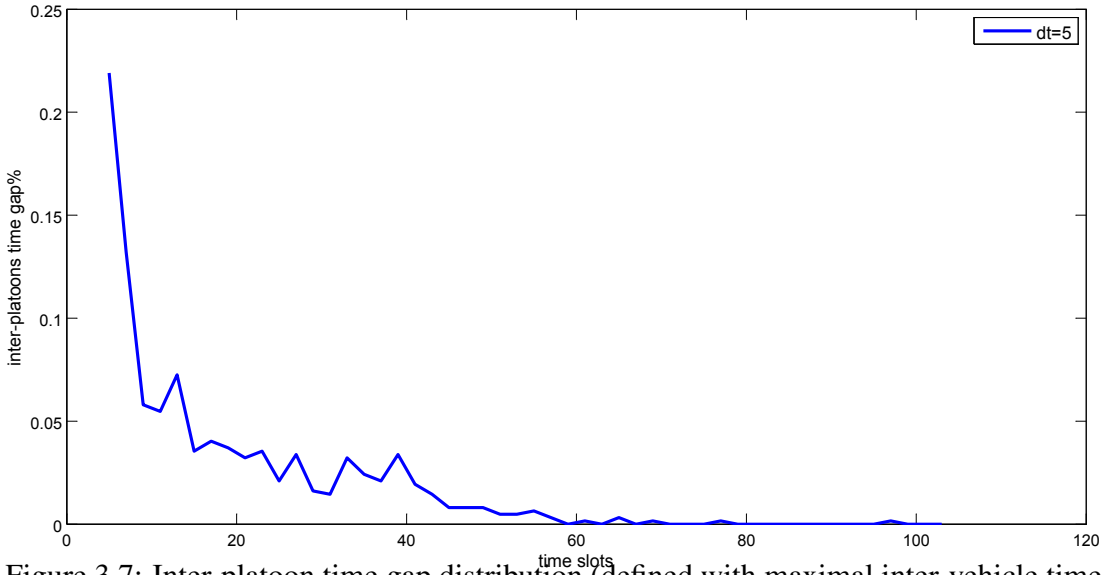


Figure 3.7: Inter-platoon time gap distribution (defined with maximal inter-vehicle time $\Delta = 5[\text{sec}]$) measured over 2 hours.

The probability distribution of the inter-platoon time gaps measured at the same location, during 2 hours, and with a $\Delta = 5[\text{sec}]$ is presented in Figure 3.7. The distribution looks consistent with a Poisson "like" distribution. The 90% of the inter-platoon time gaps are between 5 to 20 $[\text{sec}]$ which is in agreement with our choice for $\Delta = 5[\text{sec}]$. We performed this analysis at different locations (including near roundabouts, which one could think might destroy platoons).

Based on these statistics and the above arguments for using platoons presented before, we formally define a platoon as follows.

Definition 3.3. *Let T_r^k denote the time at which sensor r makes its k^{th} axle detection. Then the n^{th}* platoon*, denoted $P_{r,n}$, is defined as gathering the axle-measurements k_n to $k_{n+1} - 1$, where for each n we have*

$$T_r^k - T_r^{k-1} \leq \Delta \text{ for } k = k_n, \ldots, k_{n+1} - 1$$
$$T_r^{k_n} - T_r^{k_n - 1} > \Delta \ .$$

To obtain a reduced model of urban traffic, we characterise each platoon $P_{r,n}$ by the summarised information:

- *$T_{r,n} := T_r^{k_n}$, the time when the first axle crosses the location r;*
- *$T_{r,n,tail} := T_r^{k_{n+1}-1}$, the time when its last axle crosses the location r;*
- *$N_{r,n}$, the measured number of vehicles [1] passing location r between $T_{r,n}$ and $T_{r,n,tail}$, i.e. the size of the n-th platoon.*

Grouping vehicles into platoons yields a smaller and thus faster computationally reduced representation of traffic flow w.r.t. models based on individual vehicles (e.g. one state update per platoon with $N_{r,n}$ vehicles instead of one state update per each $N_{r,n}$ vehicles).

The motion of a platoon between sensors is modelled by assuming that all its vehicles travel at (approximately) the same speed. To reduce the model, we do not consider dispersion within platoons (see e.g. [70]). This is a good approximation on typical urban roads where vehicles are observed to move at maximal speed unless they are blocked in a queue; in fact, this makes the platoon state to remain robust along a typical urban link ($300[\mathrm{m}]$). We will later treat a queue of vehicles stopped behind a red light (associated to sensor r) as a stationary "fictitious" platoon, where $N_{r,n}$ is the size of the queue and $T_{r,n}$ is incremented continuously with the advancement of time while the queue is stopped.

[1] Axle detection, as opposed to vehicle detection, is a peculiarity of our particular sensors. For compatibility reasons with most other sensor types, we map our measurements to vehicle detections, by considering a new car-like vehicle every two axles. This is not fundamental to the algorithm's working.

3.4 The platoon based model

The platoons introduced in Section 3.3.1 are the basis of a hybrid model for urban traffic, which we describe as a discrete event system (see also [55]). We define four network components : *sources, sections, intersections,* and *sinks*. The queues themselves (one per section) are modelled as HIOA that interact with these network components (see Section 3.4.6). In our platoon based model, we describe how the platoons evolve through the components of this network.

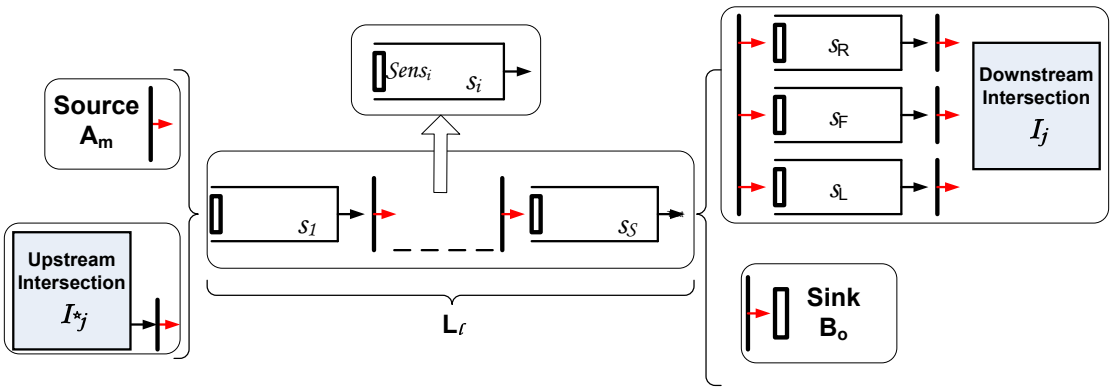


Figure 3.8: Components of an urban traffic network.

We illustrate in Figure 3.8 how the components of an urban traffic network are interconnected. Any component of the network has at least one entrance (marked by a black arrow in Figure 3.8) and/or one exit (marked by a red arrow in Figure 3.8). The interconnection between components is made by connecting the exit of component comp_1 to the entrance of component comp_2 and represented by the vertical thick lines in Figure 3.8. We say that comp_1 is upstream of comp_2 ($\text{comp}_1 = {}^*\text{comp}_2$) and comp_2 is downstream of comp_1 ($\text{comp}_2 = \text{comp}_{1^*}$). An upstream component can be interconnected to multiple downstream components (e.g. preselection lanes of an intersection). Figure 3.8 illustrates how the elements are connected to form an urban traffic network. Each component is a timed I/O automaton with the exit of the upstream component forcing arrivals at the entrance of the downstream component. Next, each component of the network will be described.

3.4.1 Road section

A road *section* $S_i \in \mathcal{S}$ consists of:

- an inflow boundary, where platoon $P_{i,n}$ enters at event time $T_{S_i,n}$;
- an outflow boundary, generating an output event when $P_{i,n}$ leaves S_i. The outflow event for S_i corresponds to an inflow event for the component that follows S_i in the network (see below);
- a fixed capacity C_i, representing the maximum number of vehicles that it can contain (i.e. when it is filled by a single queue);
- $V_{\max,i}$, the maximum speed allowed on S_i: each platoon $P_{i,n}$ will move through S_i with an independently randomly chosen speed $v_n = q \cdot V_{\max,i}$, where $q = q_2,\ q_1,\ q_0$ with respective probabilities $p_2,\ p_1,\ p_0$ (based on empirical data we used $p_2 = 0.05$, $p_1 = 0.15$, $p_0 = 0.8$, and $q_2 = 0.8$, $q_1 = 0.9$, $q_0 = 1$, respectively). We normalise the speed to express it in averaged vehicle lengths per second. This simplification is useful for simulations that will later be used for estimation and control.

The outflow boundary of S_i may be *blocked*, later defined as $exit_i = 0$, so that platoons cannot exit it and pile up to form a *queue*, whose length at time t we note $Q_i(t)$ [veh]. The delay $\delta_{i,n} = (\, C_i - Q_i(t)\,) \,/\, v_n$ [sec] expresses how long $P_{i,n}$ will drive from the inflow boundary of S_i until it reaches the end of its queue (or the end of the section if $Q_i = 0$). The time when the event "reaching exit queue" is executed is calculated as

$$T_{Q_i,n} = T_{S_i,n} + \delta_{i,n}\,. \tag{3.4}$$

The outflow boundary of S_i can be connected to the inflow of a downstream section, (the preselection lanes of) an intersection, or a sink component, defined below. The input boundary can be connected to the outflow of an upstream section, an intersection, or a source component. We denote the component connected to the outflow (resp. inflow) boundary of S_i by S_i^* (resp. ${}^*S_i$). The arrival time and size of a platoon $P_{i,n}$ that enters S_i are inherited from ${}^*S_i$, and similarly a leaving platoon is passed with its characteristic size and leaving time to S_i^*. If S_i is connected to the preselection lanes of

an intersection (parallel sections upstream an intersection), the $N_{i,n}$ vehicles of platoon $P_{i,n}$ are randomly distributed between the lanes. When $Q_i(t) = C_i$, the section is full and the outflow of its upstream component gets blocked (this is called a *spill-back*). The blocked component S_i sends a signal to the upstream component $S_g = {}^*S_i$ setting $exit_g = 0$.

If a platoon $P_{i,n}$ is moving faster than its predecessor $P_{i,n-1}$, then the head of $P_{i,n}$ could catch up with the tail of $P_{i,n-1}$. To keep things reasonably simple, we consider that in this case the platoons will be merged at the outflow boundary of S_i. The modularity of the model allows to add more details like overtaking or splitting along a section if necessary. Further, if there are parking areas in section S_i or minor side streets without sensors, then we can model

$$N_{i,n} = N_{i-1,n} + N_{enter,i,n} - N_{leave,i,n}, \tag{3.5}$$

where $N_{enter,i,n}$ and $N_{leave,i,n}$ are random numbers of vehicles that join respectively leave the platoon along S_i.

3.4.2 Source

A *traffic source* $A_m \in \mathcal{M}$ generates new platoons of random size and at random times. It is connected to the inflow of a section $S_i \in \mathcal{S}$, i.e. $A_m = {}^*S_i$. The n-th platoon is generated by A_m at time

$$T_{A_m,n} = T_{A_m,n-1,\text{tail}} + \Delta + G_{A_m,n},$$

where the time gaps $G_{A_m,n}$ are generated according to a independent exponential distribution. The platoon size $N_{A_m,n}$ is drawn from a time-dependent probability distribution

$$\text{Proba}(N_{A_m,n} = k) = p_{A_m}(k,t), \quad k = 1, 2, \ldots, K_{max}(t)$$

(built to match empirical data describing daily variations in traffic intensity). The duration $T_{A_m,n,\text{tail}} - T_{A_m,n}$ of the n-th platoon is then drawn as the sum of $N_{A_m,n}$ uniformly distributed random variables D_k in $[D^{min}, D^{max}]$, representing inter-vehicle

time gap within the platoon, with mean value $\mathbb{E}[D_k] =: M_h$.

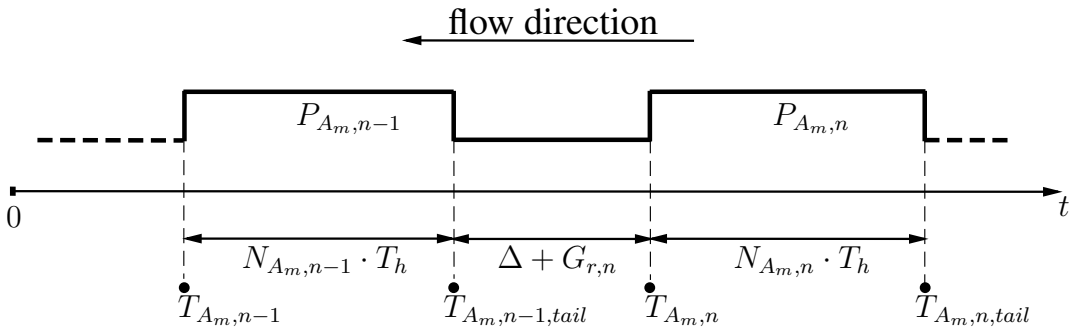


Figure 3.9: Consecutive platoons $P_{A_m,n-1}$ and $P_{A_m,n}$ emerging at the location A_m

The source parameters, presented in Figure 3.9, determine the time-varying expected emergence rate of vehicles at the location A_m,

$$\lambda_{A_m}(t) = \frac{\mathbb{E}[N_{A_m}(t)]}{\Delta + \mathbb{E}[G_{A_m}] + \mathbb{E}[N_{A_m}(t)] \cdot M_h} \text{ [veh/sec]} . \tag{3.6}$$

Here $\mathbb{E}[N_{A_m}]$ and $\mathbb{E}[G_{A_m}]$ denote the expected values of the distributions for $N_{A_m,n}$ and $G_{A_m,n}$ respectively, adjusted to match historical measurements at the location of A_m. $N_{A_m,n}$ is the number of vehicles from platoon n emerging at location A_m, with an average inter-vehicle time gap M_h, and with a time gap with the tail of the previous platoon $\Delta + G_{A_m,n}$.

3.4.3 Sink

A *sink* $B_m \in \mathcal{O}$ is just a recipient for platoons that leave the modelled network at the outflow of some section S_i, i.e. $B_m \subset S_i^*$. This component only serves to keep track of exit times for performance evaluation: a platoon $P_{m,n}$ crossing B_m is assigned a leaving time interval $[T_{B_m,n}, T_{B_m,n,\text{tail}}]$, after which it undergoes no further changes.

3.4.4 Intersection

An *intersection* $I_j \in \mathcal{I}$ is characterised by the 5-tuple

$$I_j = (In_j, Ext_j, OD_j, TLPhase_j, traf\_light_j(t))$$

where

- In_j is a list of all entrance points of intersection I_j (exit points of upstream sections);
- Ext_j is a list of all exit points of intersection I_j (entrance point of downstream sections);
- $OD_j \subseteq In_j \times Ext_j$ is a list of all entrance/exit pairs of I_j between which a traffic flow is admissible;
- $TLPhase_j$ enumerates all the possible phases of the traffic light at I_j, for each phase listing which traffic flows $\in OD_j$ it enables;
- and $traf\_light_j(t) \in TLPhase_j$ is the current state (phase) of the traffic light.

Each entrance point of an intersection can be subdivided in preselection lanes, indexed by $z \in \{$left-turning, forward, right-turning,$\dots\}$. When a platoon $P_{i,n}$ reaches the outflow boundary of a section S_i connected to I_j, the $N_{i,n}$ vehicles from $P_{i,n}$ are randomly distributed among the preselection lanes of the corresponding entrance point according to a multinomial distribution $p_z(t)$ determined from historical data at I_j.

If the priority and safety rules are satisfied (green traffic light, right-of-way rules for left turning traffic are satisfied, downstream section is not blocked), then the platoon $P_{z,n}$ now assigned to lane z moves to the associated exit point in Ext_j, corresponding the OD_j-pair the platoon belongs to. This motion takes a random time δ_{I_j}, determined from historical data and possibly depending on road conditions. If the green period is too short for the complete platoon to pass, then $P_{z,n}$ is split and the size of the passing platoon is determined by the length of the green period.

3.4.5 Hierarchical model

In order to set up a hierarchical model structure for large networks, we introduce the following intermediate element.

A *link* $L_l \in \mathcal{L}$ is a sequence of sections $S_i \in \mathcal{S}$ glued together, such that a proper renumbering yields $S_i = {}^*S_{i+1}$ for all S_i, ${}^*S_{i+1} \in L_l$.

All the sections forming the link L_l use the same maximum speed V_{max}. The subdivision of links into sections has been introduced such that *each section inflow boundary is associated with a sensor.*

Combining the enumerated components leads to a reduced model of urban traffic, which allows to reasonably simulate networks of moderate size.

Definition 3.4. *An* urban traffic network, $\aleph$, *consists of a set of links (roads)* $L_l \in \mathcal{L}$, *connecting intersections* $I_j \in \mathcal{I}$, *sources* $A_m \in \mathcal{M}$, *and sinks* $B_o \in \mathcal{O}$. *The* topology $\mathcal{T}(\aleph)$ *of* $\aleph$ *is characterised by specifying, for each intersection* $I_j \in \mathcal{I}$,

- *the links* $\in {}^*I_j$ *carrying traffic towards each of its entrance points* $\in In_j$, *and*
- *the links* $\in I_j^*$ *carrying traffic leaving each of its exit points* $\in Ext_j$.

Each "free" link inflow/outflow boundary is then connected to a source/sink.

Definition 3.5. *The* state $X_{S_i}(t)$ *of section* S_i *consists of*

- *a list, with their properties, of all the platoons inside* S_i *at time* t*; i.e. for all* $P_{i,n}$ *such that* $T_{S_i,n} \leq t \leq T_{S_i^*,n,tail}$, *the state remembers* $(T_{i,n}, N_{i,n})$*;*
- *the size of the queue* $Q_i(t)$ *at the downstream boundary of* S_i.

The state of the link L_m is obtained by stacking the states $X_{s_i}(t)$ of all the $S_i \subset L_l$.

Definition 3.6. *The state of the network* $\aleph$ *is obtained by stacking the state of its components, and can equivalently be written as the tuple:*

$$X_{\aleph} = (P_m,\ \mathcal{I})\,, \tag{3.7}$$

where $P_m(t) = \bigcup_{S_i \in \mathcal{S}} X_{S_i}(t)$ *is the set of platoons (including queues) present in the network at time* t, $\mathcal{I} = \{I_j\}$ *and* I_j *includes all the properties of intersection* j, *among others the state of the automaton representing its traffic light* $traf\_light_j(t)$.

Traffic lights and axle-detecting sensors are respectively the inputs and outputs with which $\aleph$ will be controlled. The control aspects of the network $\aleph$ are detailed in Sections 4 and 5.

3.4.6 Hybrid model

The dynamics of a queue at the outflow boundary of a section or in the preselection lanes of an intersection, are described by the I/O automaton shown on Figure 3.10. The input events are the arrival of a new platoon from the upstream component (e.g. ${}^*S_i$) and the full queue message sent by the downstream section S_i^*. Furthermore, the output events are represented by the departure of a platoon (that generates the arrival of a new platoon in the downstream component S_i^*) and the message sent to the upstream component ${}^*S_i$ when the queue is full blocking the exit of ${}^*S_i$.

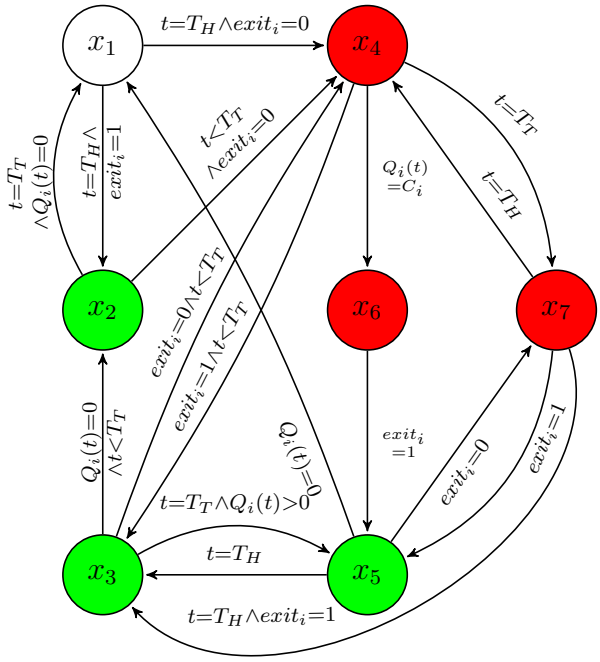


Figure 3.10: The timed I/O automaton describing a queue: red states have $exit_i = 0$ (outflow blocked) and green states $exit_i = 1$ (outflow open), while for x_1 the condition of $exit_i$ does not matter.

The state of the outflow boundary of section S_i is represented by $exit_i \in \{blocked = 0, free = 1\}$. An $exit_i = 0$ can occur if the downstream component reaches its maximal queue length ($Q_i(t) = C_i$) or if the downstream traffic light is red. Figure 3.10 further uses the generic notation T_H, T_T respectively for $T_{Q_i,n}$ and $T_{Q_i,n,\text{tail}}$ of a particular platoon. The guards, indicated along the arrows, specify the jump conditions that force

an event, i.e. a state transition, to take place:

- In state x_1: $Q_i(t) = 0$, no approaching platoon. When a platoon arrives and the outflow is not blocked ($t = T_H \wedge exit_i = 1$) there is a transition to state x_2; if a platoon arrives and the outflow is blocked ($t = T_H \wedge exit_i = 0$), a transition to x_4 occurs;

- In state $x_2 : Q_i(t) = 0,\ exit_i = 1$ and a platoon is currently crossing location i ($\exists\, P_{i,n} \,:\, T_H \leq t < T_T$). As soon as the platoon has passed ($t = T_T \wedge Q_i(t) = 0$), there is a transition back to x_1; if the exit becomes blocked while there are still vehicles to pass ($exit_i = 0 \wedge t < T_T$), a transition to x_4 takes place.

- In state $x_3 : 0 < Q_i(t) \leq C_i$, a platoon is currently arriving ($\exists\, P_{i,n} \,:\, T_H \leq t < T_T$), and $exit_i = 1$. If the downstream component (section S_i^* or traffic light) sends a message that the exit becomes blocked ($exit_i = 0$), and there are still vehicles to pass $t < T_T$, then a transition to x_4 takes place; else if the tail of the arriving platoon $P_{i,n}$ reaches the queue ($t = T_T$) there is a transition to x_5; else if the queue becomes empty ($Q_i(t) = 0$), but the arriving platoon has not completely passed ($t < T_T$), then there is a transition to x_2.

- In state $x_4 : Q_i(t) > 0$, a platoon is currently arriving ($\exists\, P_{i,n} \,:\, T_H \leq t < T_T$), and $exit_i = 0$. If the queue reaches the maximum capacity ($Q_i(t) = C_i$), a transition to x_6 occurs; if $Q_i(t) < C_i$ and the arriving platoon merges completely with the queue ($t = T_T$), transition to x_7 takes place. If the exit becomes unblocked ($exit_i = 1$), there is a transition to x_3.

- In state x_5: $0 < Q_i(t) \leq C_i$, $exit_i = 1$ and the queue is depleting without new arrivals. If $Q_i(t)$ becomes empty, the state jumps to x_1; if instead the exit becomes blocked ($exit_i = 0$), then transition to x_7 occurs; if a new platoon starts to join the non-empty queue ($t = T_H$), there is a state transition to x_3.

- In state $x_6 : exit_i = 0$ and the queue is full ($Q_i(t) = C_i$), blocking the exit of the upstream section. Whenever a transition to/from x_6 occurs a message must be sent to the upstream component indicating that its exit has become blocked/unblocked. The only transition allowed from x_6 is jumping to x_5 when the exit of S_i gets unblocked.

- In state x_7 : $exit_i = 0$, $0 < Q_i(t) \leq C_i$ and there is no arriving platoon. If a platoon starts to merge with the queue ($t = T_H$ for some $P_{i,n}$) then a transition to x_4 occurs; if $exit_i$ is unblocked, a transition to x_5 occurs.

The automaton associated to a queue in each component i exchanges messages with the automata of upstream and downstream components ${}^*i$ and i^*. As a platoon passes from i to i^*, its characteristic properties are transferred between the corresponding automata. Transitions involving state x_6 of component i are particular, associating component ${}^*i$ as follows. When $Q_i = C_i$ and $exit_i$ becomes unblocked, i switches to x_5 and sends an "unblocked" message to ${}^*i$. If $Q_{*i} = 0$, nothing more happens. But often, a queue $Q_{*i} > 0$ will have built up behind the blocked outflow. Then at reception of the "unblocked" message, the automaton of ${}^*i$ jumps to x_3 or x_5 and it sends a platoon to i, whose automaton jumps to x_3. Then the evolution of Q_{*i} will depend on the speed at which the platoon leaves ${}^*i$ (i.e. can enter i); so i must send this speed information to ${}^*i$, which itself will communicate the properties of the transferred platoon to i.

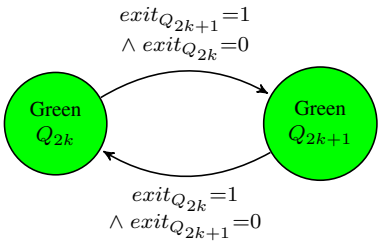


Figure 3.11: The timed input/output automaton describing the evolution of a traffic light.

The evolution of a traffic light is modelled by the simple automaton shown in Figure 3.11. Each state represents a green period for one set of non-conflicting queues ($\equiv OD$-pairs), e.g. $Q_{2k} = \{$North-South, South-North traffic$\}$ and $Q_{2k} = \{$East-West, West-East traffic$\}$. The switching can be induced by an internal clock or by specific events generated by the other automata. More complicated traffic lights, optimizing the switching process with orange and 'buffer' phases, can be represented similarly [54].

3.5 Implementation and performance

The platoon based model described above can be efficiently implemented in a discrete-event system simulation tool. The discrete-event simulation framework and its components are introduced in Section 2.3. The agenda keeps track of all the future events: *all transitions in queue automata*, including *platoon-enters-intersection, platoon-enters-link, platoon-leaves-intersection, platoon-leaves-link*; all *traffic-light-switches*; and *platoon-generated-by-source*.

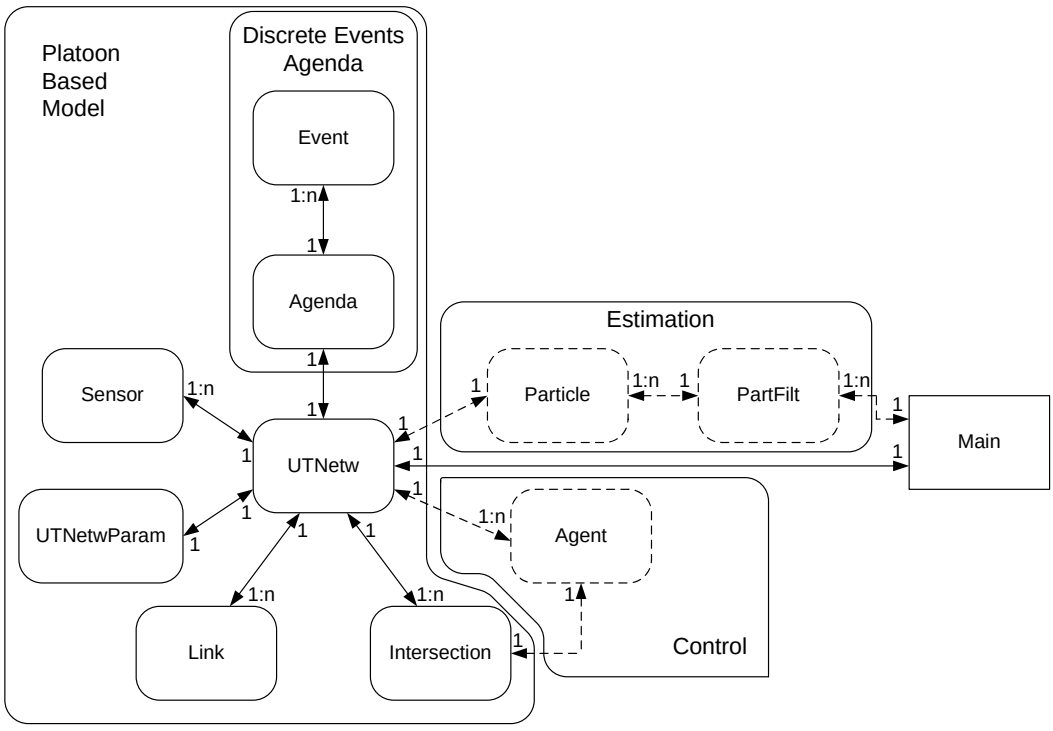


Figure 3.12: Structure diagram - dotted line represents the objects used the next chapters for estimation and control.

Each event has the following elements:

- t_event, the time at which it occurs;
- $action\_type$, the type of event (e.g. platoon arriving);

- $place$, the list of locations (components of $\aleph$) affected;
- $general\_purpose$, a parameter used for different purposes depending on $action\_type$ (e.g. platoon characteristics or duration of next green period).

The model is currently coded in MATLAB using the Object-Oriented Programming (OOP) paradigm [65]. The entire OOP program is a collection of interacting objects where each object is capable of receiving messages, processing data, and sending messages to other objects. We briefly describe the objects and the relations between them. Figure 3.12 depicts the objects and their relations, represented by arrows, between the different objects . The type of relations that appear are either 1-to-1, 1-to-n, or n-to-1 (e.g. a 1-to-n relation shows that to 1 object from the left corresponds n objects from he right where n is generic).

The *UTNetw* class, used as container to describe the urban network $\aleph$, encapsulates the list of links, the list of intersections, and the list of sensors. The general parameters of the urban network are contained in *UTNetwParam* object. Each *Link* object contains the index of the upstream and downstream intersections or source/sink along with the other attributes that describe a link. An *intersection* object contains the list of all the links or source events connected to its entrance points together with the list of all the downstream links or sinks connected to its exit points, current green direction, current green length, and other internal parameters. The list of *Sensor* objects contains the locations where the sensors are present in the network. Since the model is implemented as a discrete event simulator, the agenda object contains the future events that may have to be executed. Each *event* object contains the time when the event will happen, a general purpose variable (e.g. platoon characteristics, next green length), place where the event takes place, and the type of action of the event (e.g switch green, source event, sink event, etc.). Currently the configuration of a network is done manually but a configuration tool can be developed to automatically generate such configuration files for initialisation. The inflow traffic of a given network can be either generated on-line (after the arrival of a platoon a new platoon is generated) or can be loaded from a pre-generated file (for replication of results).

In Figure 3.12 are also presented modules used later for doing estimation and control. The ”main” block is the wrapper file that is calling the initialisation file(s) and is running the simulation.

The reasons for using OOP and MATLAB are for keeping the flexibility during the entire development cycle. As a first prototype the computation speed of the implementation was not the main goal. A compiled implementation (e.g. C++) could significantly increase the computational speed. The described structure of objects can be fully reused in the further development of a faster more optimised version.

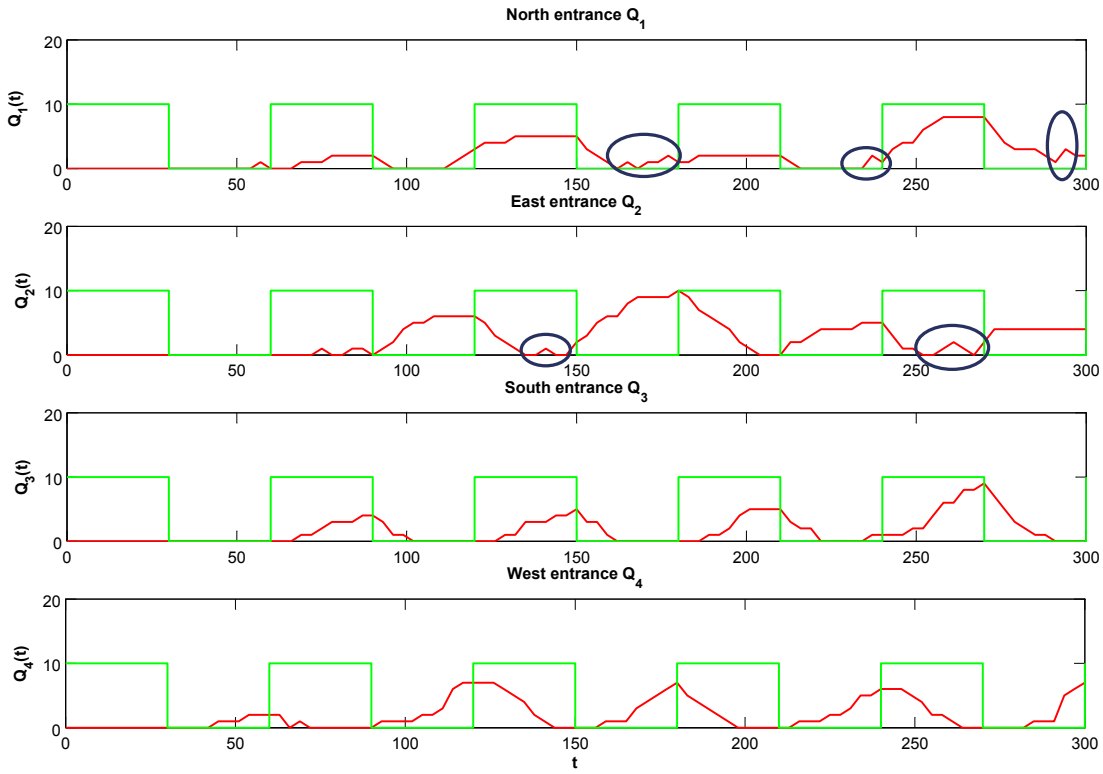


Figure 3.13: The queues formed at a typical intersection. The queues are in red and the phases of the traffic light are presented in green (sampling error due to the approximation made by the sensors).

An example of the queues $Q_i^j(t)$ (as shown in Figure 3.2 and with j index dropped) formed in front of the traffic light at a typical intersection using the platoon based model is presented in Figure 3.13. The initial condition of the simulation is with an empty network and predefined traffic light switches at every $30[\text{sec}]$. The scale is for the red curve that indicates the number of vehicles. The green curve depicts the traffic light

phases (lower value North-South is green, and higher value East-West is green). This run is used only to illustrate that the simulator correctly represents the traffic behaviour. Different kinds of checks (e.g. following the platoons evolution throughout a network, vehicles conservation) are performed to validate the implementation.

Note that the number of cars in a preselection lane queue can occasionally increase during green (marked by ovals in Figure 3.13), when a car arrives at the end of the preselection lane while no car is immediately ready to leave into the intersection. Of course this can only happen with very few vehicles, since when a vehicle has arrived in the queue, it will be ready to leave the queue at the next sampling time.

The computational complexity of the platoon based model is proportional to the number of events to be handled (much smaller than the number of vehicles). The number of events is independent of traffic intensity (size of platoons grows with the traffic intensity whereas the number of platoons remains approximately unchanged).

3.6 Summary

The heterogeneity of the urban traffic (the gaps between platoons of vehicles) is captured in a computationally efficient way by the platoon based model. Using the platoons as an abstraction of the real traffic the executed number of events by the discrete event model is reduced - instead of having an event for each vehicle we have an event for each platoon present in the urban network.

Chapter 4

Distributed estimation of urban traffic networks

First we explain why state estimation is needed for feedback control of traffic lights as proposed in [54] and in Chapter 5. Then we show how to perform this estimation with a particle filter based on the model of Section 3.4. Section 4.2 presents a standard particle filter. Given the size of the network, data collection and control decisions are not usually concentrated in one single large traffic control centre. Thus we need to develop distributed estimators of the traffic state. Results presented in Section 4.3 show how a distributed particle filter implementation enables application to large systems.

4.1 Issues for estimation

Urban traffic state estimation based on measurements faces several difficulties. First, the inverse problem — retrieving the most likely state that has *caused* the observations — admits a large variety of plausible a priori solutions, e.g. a low measured traffic flow can be equally due to few vehicles (low density of traffic) or to a very low speed of a dense queue. Moreover, the queue size at an intersection is the *integrated* difference between noisily observed inflow and outflow, and integrating noise leads to a random walk that can quickly diverge. It is therefore crucial to improve estimates by taking into account the relationship between successive sensors and modelled dynamics, knowing the state of the traffic lights. For instance, the outflow immediately after a traffic light

turns green certainly reflects the queue-size at its inflow at the end of the red phase; this dependence allows to correct previous queue size estimations and alleviates among others the two issues just mentioned. Similarly, a large platoon estimated from flows at an upstream sensor should be confirmed by observing a high traffic flow a short time later at a downstream sensor. What we are looking for is to efficiently combine all this information to build queue size and platoon location estimations that are coherent with the expected dynamics of traffic behaviour.

At several locations in the network (including upstream and downstream of each intersection and at all section boundaries), sensors detect the successive passage times of vehicles. However, these sensors are very noisy, often failing to detect passing vehicles, and sometimes randomly generating output when no vehicles are passing. Sensor outputs can therefore not be used directly for queue and traffic flow estimation. Instead, a recursive filter must be used that combines available online data with constraints imposed by a model of reasonably expected traffic behaviour e.g. consequences of the known red/green switching times of traffic lights, and relations between passage times at successive locations along the same road.

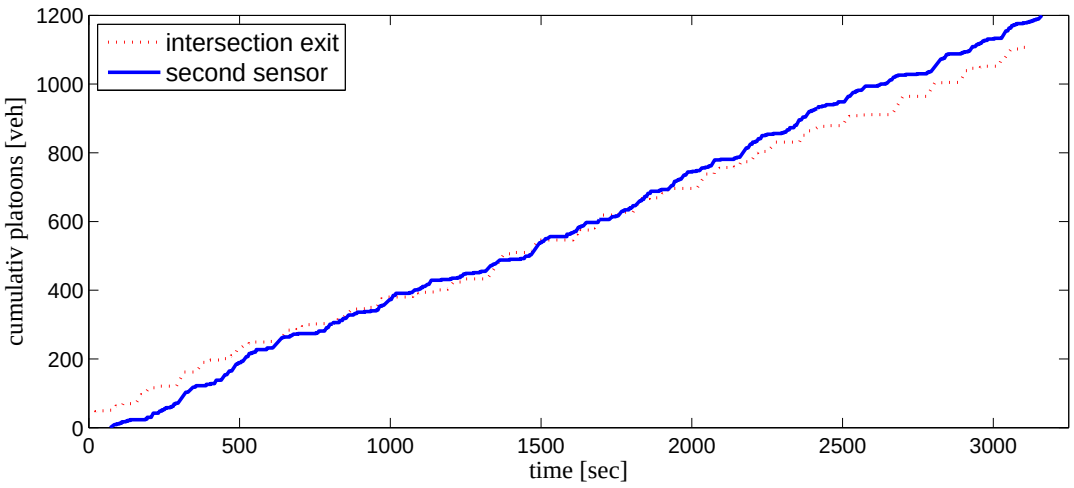


Figure 4.1: Cumulative flow at two consecutive sensor locations on the same road. For a better comparison, the curve of the second sensor is shifted in time according to the estimated time delay $=$ (distance ≈ 1 $[\mathrm{km}]$ divided by mean-vehicle-speed ≈ 70 $[\mathrm{km/h}]$) between the two sensors.

To investigate the sensor accuracy, using the data already presented in Section 3.3.1, we have compared the cumulative flow at two consecutive sensor locations ($\approx 1[\mathrm{km}]$ apart) on a long stretch of road without intersections, parking areas or side roads (Figure 4.1). Ideally, the flow should be conserved, i.e. the number of vehicles that pass the first sensor should be preserved $1[\mathrm{km}]$ downstream at the second sensor location. In the data the flow is apparently not conserved.

4.1.1 Sensor model

Before going on, let us specify our sensing model. We place a sensor at each section inflow boundary and at each selection lane of intersection inflows. Denoting by Δ_s the duration of a communication round in real-time traffic management, a sensor is typically modelled as giving noisy estimates about the number of vehicles $M_{r,k}$ that cross location r during each time interval $[(k-1)\,\Delta_s\,,\ k\Delta_s]$, with k integer.

More precisely, the sensor output

$$y_{r,k} = M'_{r,k} + m_r(k) \tag{4.1}$$

represents two effects. Missed detections are taken into account by making $M'_{r,k}$ a binomial random variable on $M_{r,k}$ trials, each with $p_r = 0.95$ probability of success. And false detections are represented with a distribution: $m_r(k) = 0,\ 1$ or 2 with respective probabilities 0.95, 0.04 and 0.01. All these random events are considered independent in time and among sensors, and their characteristics can be adjusted from specifications or from a statistical data analysis.

The sensor noise inevitably implies that there will always be an uncorrelated uncertainty in their signals, such that the estimated cumulative flow(s) — similar to a queue-length — diverge from the real situation like a classical random walk - stochastic process. In fact, the noise in current traffic sensors is so large that just applying low-pass filtering and conservation laws yields too large errors in queue size estimations or in estimates of platoons in sections.

4.1.2 Arguments for using a particle filter

To get viable traffic estimates, we must therefore use a Bayesian recursive filter that confronts the data measured online with predictions from a system and sensors' model: this adds a causal relation to the estimation, by taking into account how the observed behaviour must be a plausible *consequence* of the previously estimated state.

In [77] recursive traffic state estimation is achieved via a Kalman filter (KF) combining sensor data with conservation equations of vehicles. Indeed when working on a car-by-car basis, the nonlinearity of the dynamics remains low enough to allow good results with an Extended Kalman filter (EKF). However, in large networks the dimension of a car-by-car state becomes too large to use an EKF for online control. In this case a particle filter (PF) can be a faster option. Furthermore, the real traffic sensors that we tested introduce strongly non-Gaussian perturbations (missed cars, false detections). In fact, their limited accuracy justifies the use of a reduced model, i.e. our platoon-based proposal, that however further increases nonlinearity, but drastically reduces the state dimension towards computations feasible in real-time. The increased nonlinearity would make an EKF poorly robust, but does not affect the PF which fully benefits from the reduced dimension.

Therefore, given the complexity of the model, a particle filter estimator is the method of choice, because it only requires to simulate a large number of random evolution possibilities. For those simulations, we propose to use the PBM as a computationally efficient model describing those variables (platoon location, queue size) that need to be estimated. This enables fast computations without losing details that are important for control (like arrival times of platoons, which would be absent from e.g. a time-averaged fluid flow model). The PF has already been used for motorway traffic estimation [4, 58, 59]; we proposed to adapt it for urban traffic as well [53, 56].

For large networks, a centralised PF implementation would still need very large capabilities in terms of computation and communication as well as robustness of the IT network. Fortunately, the modularity of the PBM and of its discrete event simulation implementation allows to develop a distributed PF algorithm which requires little communication between local agents: only arrival times and sizes of platoons must be exchanged at the boundaries of directly connected components.

4.2 Standard particle filter

The particle filter, also known as a sequential Monte Carlo Method, is a model based state estimation technique based on simulations. The first particle filter algorithms were suggested by [80] and [35]. A good and complete introduction to particle filters can be found in [26].

Given any general non-linear, stochastic, discrete-time state evolution

$$x_k = f_k\left(x_{k-1}, u_{k-1}, v_{k-1}\right), \tag{4.2}$$

where x_{k-1}, x_k are previous and current states, u_{k-1} is the known input of the system in the interval of time $[(k-1)\,\Delta_s\,,\ k\Delta_s)$, and v_{k-1} is white noise (usually non-Gaussian, uncorrelated for different values of k) affecting the system. Since new measurements are collected every Δ_s, we translate the PBM into discrete time. The time discretisation is simply obtained from the output of the simulator, described in Chapter 3, every Δ_s time units.

The online measurement at time $k\Delta_s$ is given by

$$y_k = h_k\left(x_{\tau\in[(k-1)\Delta_s\,,k\Delta_s)}, e_k\right), \tag{4.3}$$

where e_k is measurement white noise (i.e. independent for different values of k). The sensor model used in this thesis is described in Section 4.1.1. Our goal is to estimate x_k from all the measurements up to time $k\Delta_s$. Since (4.2), (4.3) are stochastic, x_k cannot be exactly determined from the measurements. The best characterisation of x_k is $p\left(x_k|y_{1:k}\right)$, its conditional probability distribution (cpd) given all the past measurements $y_{1:k} = [y_1\ y_2\ \ldots\ y_k]$. A recursive filter can update this cpd each time a new sensor measurement is processed by the Bayes law, as follows.

- Prediction step: taking the estimation $p\left(x_{k-1}|y_{1:k-1}\right)$ at step $k-1$ as a distribution of initial conditions, compute their likely evolution under $f_k(.)$:

$$p\left(x_k|y_{1:k-1}\right) = \int p\left(x_k|x_{k-1}\right) p\left(x_{k-1}|y_{1:k-1}\right) \mathrm{d}x_{k-1}\,. \tag{4.4}$$

  The state transition probability distribution function (pdf) $p\left(x_k|x_{k-1}\right)$ is just a reformulation of $f_k(.)$.

- Measurement update step: correct the predicted estimation by multiplying the likelihood of state values by their likelihood to generate the observed measurement value y_k according to $h_k(.)$:

$$p\left(x_k|y_{1:k}\right) = \frac{p\left(y_k|x_k\right) p\left(x_k|y_{1:k-1}\right)}{\int p\left(y_k|x_k\right) p\left(x_k|y_{1:k-1}\right) \mathrm{d}x_k} \,. \tag{4.5}$$

  The measurement likelihood pdf $p\left(y_k|x_k\right)$ is a reformulation of $h_k(.)$ from the model.

An exact implementation of this Bayesian recursive filter is almost always intractable, because efficient numerical representations of the full cpd only exist for particular, a priori known distribution types, and the integration in (4.4) and (4.5) is usually numerically infeasible. (The KF precisely takes advantage of the fact that linear systems, with Gaussian additive noise, preserve a Gaussian distribution type for the cpd). The PF (see e.g. [27, 71] for tutorials) builds on a Monte Carlo representation of the integrals in the cpd formula, i.e. it uses a fixed number $\mathcal{N}_\mathcal{P}$ of random samples of the state x_k^i, with associated weights w_k^i ($\sum_{i=1}^{\mathcal{N}_\mathcal{P}} w_k^i = 1$), such that $\sum_{i=1}^{\mathcal{N}_\mathcal{P}} w_k^i \, g(x_k^i)$ approximates $\int p\left(x_k|y_{1:k}\right) g(x_k) \, \mathrm{d}x_k$ as well as possible for any useful function $g(.)$. Thus each pair $\langle x_k^i, w_k^i \rangle$, called a particle, represents a hypothesis of the real state trajectory of x_k, with an associated belief w_k^i.

The key to computing the weights is the importance sampling principle. Consider a known proposal pdf $q(x)$, from which $\mathcal{N}_\mathcal{P}$ samples x^i are drawn. Then by taking

$$w^i \propto \frac{p(x^i)}{q(x^i)}, \tag{4.6}$$

the associated empirical histogram of $\langle x^i, w^i \rangle$ will approximate the pdf $p(x)$ with high probability for sufficiently large $\mathcal{N}_\mathcal{P}$. From (4.6) it is clear that, for given $\langle x^i, w^i \rangle$ representing $p(x)$, a valid representation of some other distribution $p'(x)$ is obtained by keeping the x^i and adapting the weights to

$$w'^i \propto \frac{p'(x^i)}{q(x^i)} = \frac{p'(x^i)}{p(x^i)} \, w^i \,. \tag{4.7}$$

This principle directly fits the needs of recursive state estimation. Starting with a set

$\mathcal{P}$ of particles supposed to represent $p\left(x_{k-1}|y_{1:k-1}\right)$, it suffices to execute a stochastic simulation of the model (4.2) over the time interval $[(k-1)\Delta_s,\ k\Delta_s)$ for each x^i and leave the w^i unchanged, to get a Monte Carlo representation of $p\left(x_k|y_{1:k-1}\right)$ (prediction step).

The measurement update step then takes advantage of the special form of (4.5) to implement efficient importance sampling: keeping the same x_i, the correction from $p(x) = p\left(x_k|y_{1:k-1}\right)$ to the wanted $p'(x) \propto p(x)p(y_k|x_k^i)$ simplifies to

$$w_k^i \propto w_{k-1}^i\, p(y_k|x_k^i)\ . \tag{4.8}$$

A well-known problem of PFs in practice is *degeneracy* [5, 25]. As the x^i evenly explore all a priori possible evolution cases, many particles — associated to options that are unlikely to have taken place according to measurements — will have a weight w^i close to 0 after a few iterations; those particles convey little relevant information about the situation, such that the effective particle set for exploring future evolution is reduced to a number $\ll \mathcal{N}_\mathcal{P}$. The solution is to resample the particle states on a regular basis: particles with small weights are replaced by newly created particles with states x^i close to more likely situations, after which all weights w^i are readjusted. There exist many different resampling algorithms and methods to determine when resampling is necessary. After exploring a few possibilities for our case, we decided to perform a deterministic resampling method at each iteration, by duplicating the most likely half of the particles, *with their weight*, and dropping the other half. This method keeps many likely particle options for future evaluation, like traditional stochastic resampling. But in contrast, it also keeps significant weight differences among particles that reflect their whole history; this facilitates interpretation of the results, allowing to evaluate possibilities in terms of individual particle weights instead of necessitating heavy particle density computations.

Particle filter for urban traffic

Consider an urban traffic network $\aleph$ as introduced in Section 3.4, with $\mathcal{N}_{sens}$ sensors at the inflows of sections and preselection lanes. The measurement vector, $y_k = \left[y_k^1\ y_k^2\ \ldots\ y_k^{\mathcal{N}_{sens}}\right]$, contains the number of vehicles that pass by each sensor in the time interval $[(k-1)\,\Delta_s\,,\ k\Delta_s)$. The full state of the network, $X_\aleph$, is unknown. Based on

the observations y_k, we want to determine the queue lengths in front of each intersection and the location and size of all platoons inside the network $\aleph$. Algorithm 4.1 describes how we apply the just presented PF for this task.

At time $k = 0$, we generate $\mathcal{N}_\mathcal{P}$ random particle states $X^i_\aleph$, each one including sizes and locations of platoons throughout the network and being assigned a weight $w^i_0 = 1/\mathcal{N}_\mathcal{P}$. After this initialisation, the following recursion is performed.

- The prediction step (line 7 in Algorithm 4.1) amounts to running the DES simulator of the PBM $\mathcal{N}_\mathcal{P}$ times in parallel to generate for each particle the state evolution in the interval $[(k-1)\Delta_s,\ k\Delta_s)$.
- When the information y_k is received from the sensors, the update step (line 10 in Algorithm 4.1) multiplies the weight of each particle by $p\left(y_k|x^i_k\right)$, the likelihood that it would give the observed measurement.
- The weights are renormalised; this step can also be delayed until after resampling.
- The state of the particle(s) with the highest weight(s) (or of the k particle) is sent to the output as being the estimated state $\hat{X}_\aleph$. This choice has the advantage to work without assuming any structure on the state space and probability distribution.
- Resampling (lines 18-22 in Algorithm 4.1) duplicates the $\mathcal{N}_P/2$ most likely particles and throws away the $\mathcal{N}_P/2$ least likely ones.

The particle filter replaces the very complicated explicit formula for the transition probability distribution over high-dimensional $X^i_\aleph$ required by the Bayesian filter, by $\mathcal{N}_\mathcal{P}$ randomly simulated Monte Carlo samples. The larger $\mathcal{N}_\mathcal{P}$, the better the empirical histogram generated by the particles will approximate the true cdf $p\left(X_\aleph(k)|y_k\right)$. The speed of the simulation step — still the most demanding one — is thus crucial to ensure best possible performance by allowing larger $\mathcal{N}_\mathcal{P}$. The platoon-based model with DES implementation proposed in this paper is a first step towards a computationally tractable PF for urban traffic (see results in Section 4.4). To further speed up computations for large networks, we next propose a *distributed* PF implementation.

Algorithm 4.1 Centralised particle filter

Ensure: $\rightarrow$ Initialisation of the particles at $k = 0$
for $i = 1$ to $\mathcal{N}_\mathcal{P}$ **do**
  generate the i^{th} sample $\{x_0^i\}$
  initialise i^{th} weight with $w_0^i = 1/\mathcal{N}_\mathcal{P}$
end for
for all k such that $k \geq 1$ **do**
Ensure: $\rightarrow$ Prediction step
  for $i = 1$ to $\mathcal{N}_\mathcal{P}$ **do**
    propagate i^{th} sample $\{x_k^i\}$ with $p\left(x_k|x_{k-1}^i\right)$ depending on $f_k(.)$ as in (4.4)
  end for
Ensure: $\rightarrow$ Update step
  for $i = 1$ to $\mathcal{N}_\mathcal{P}$ **do**
    update the weight of the i^{th} sample according to $w_k^i = w_{k-1}^i \cdot p\left(y_k|x_k^i\right)$ as in (4.5), (4.6)
  end for
Ensure: $\rightarrow$ Normalisation
  compute new total weight $W_k = \sum_{i=1}^{\mathcal{N}_\mathcal{P}} w_k^i$
  for $i = 1$ to $\mathcal{N}_\mathcal{P}$ **do**
    replace w_k^i by the normalised w_k^i/W_k
  end for
Ensure: $\rightarrow$ Output
  order particles by their weight.
  $\hat{X}_\aleph \leftarrow$ the state of the particle(s) with the highest weight(s).
Ensure: $\rightarrow$ Resampling
  for $i = 1$ to $\mathcal{N}_\mathcal{P}/2$ **do**
    replace the state of the particle $x_k^{\mathcal{N}_\mathcal{P}-i+1}$ with the state of the particle x_k^i
    $w_k^{i_{new}} = \left(w_k^i + w_k^{\mathcal{N}_\mathcal{P}-i+1}\right)/2$
    $w_k^i = w_k^{i_{new}}$ and $w_k^{\mathcal{N}_\mathcal{P}-i+1} = w_k^{i_{new}}$
  end for
Ensure: $\rightarrow$ Time step update
  $k \leftarrow k + \Delta_s$
end for

4.3 Distributed particle filter

For very large networks, following all the platoons and queues can become a heavy burden for a central agent, with an overloaded DES agenda in the simulation step and an overflow of communication with all sensors. Thanks to the modularity of the PBM, a distributed implementation of the PF can solve this problem.

In a general description, we start by partitioning the state and measurement vectors into $\mathcal{N}_R$ subvectors $x^{(r)}$ and $y^{(r)}$, $r = 1, 2, \ldots, \mathcal{N}_R$, such that (4.2), (4.3) writes

$$x_k^{(r)} = f_k^{(r)}\left(x_{k-1}^{(r)}, u_{k-1}^{(r)}, b_{k-1}^{(r)}, v_{k-1}^{(r)}\right), \tag{4.9}$$

$$y_k^{(r)} = h_k^{(r)}\left(x_{\tau\in[(k-1)\Delta_s\,,\,k\Delta_s)}^{(r)}, e_k^{(r)}\right), \quad \text{for } r = 1, \ldots, \mathcal{N}_R. \tag{4.10}$$

The output of subsystem r thus only reflects its local state, but the state can be updated as function of the whole network state thanks to the vector $b_{k-1}^{(r)}$ of variables whose values are shared by communication with neighbouring subnetworks. This can of course be done in many ways — in an extreme case each vector $b_{k-1}^{(r)}$ would cover all x_k — but for networks with a spatial structure, it is reasonable to expect that a natural partitioning will allow to keep the dimension of $b_{k-1}^{(r)}$ small and independent of the network size.

It is adequate to assume that the noise affects each partition independently, such that

$$p(x_k|x_{k-1}) = \prod_{r=1}^{\mathcal{N}_R} p(x_k^{(r)}|x_{k-1}^{(r)}, u_{k-1}^{(r)}, b_{k-1}^{(r)}) \tag{4.11}$$

$$p(y_k|x_k) = \prod_{r=1}^{\mathcal{N}_R} p(y_k^{(r)}|x_k^{(r)}). \tag{4.12}$$

Then one can assign to each subnetwork r its own local particle filter $PF^{(r)}$. Each $PF^{(r)}$ has its own $\mathcal{N}_\mathcal{P}$ particles $\left\langle x_{k-1}^{(r),i}, w_{k-1}^{(r),i}\right\rangle$ and only indirectly shares, with each of its neighbours j, the part of $x_{k-1}^{(r),i}$ intervening in $b_{k-1}^{(j),i}$, which we denote $b_{k-1,r}^{(j),i}$. The message-passing ensures that the group

$$\mathcal{P}_i = \{\, \left\langle x^{(r),i}, w^{(r),i}\right\rangle \;:\; r = 1, 2, ..., \mathcal{N}_R\} \tag{4.13}$$

of all particles with the same index i in the different subnetworks, form a consistent

description for the whole traffic, with associated probability weight

$$w^i = \prod_{r=1}^{\mathcal{N}_R} w^{(r),i} \ , \ \sum_{i=1}^{\mathcal{N}_\mathcal{P}} w^i = 1 \, .$$

Before the prediction step k, a round of local message exchanges ensures that each $PF^{(r)}$ collects the information of its $b_{k-1}^{(r),i}$. Then $PF^{(r)}$ uses the latter, for each i in conjunction with $x_{k-1}^{(r),i}$, $u_{k-1}^{(r),i}$ and a random sample for the stochastic component $v_{k-1}^{(r),i}$, to generate a possible $x_k^{(r),i}$ with (4.9), much like for the centralised PF. The update step adapts the weights to complete the importance sampling Monte Carlo computation by implementing a distributed version of (4.8). The assumptions (4.10), (4.12) made about local output are fundamental here, allowing to write

$$\begin{aligned} p(y_k|x_k^i)\, w_k^i &= \left(\prod_{r=1}^{\mathcal{N}_R} p(y_k^{(r)}|x_k^{(r),i}) \right) \left(\prod_{m=1}^{\mathcal{N}_R} w_{k-1}^{(m),i} \right) \\ &= \prod_{r=1}^{\mathcal{N}_R} \left(p(y_k^{(r)}|x_k^{(r),i})\, w_{k-1}^{(r),i} \right) \ , \end{aligned}$$

where each factor on the last line can be computed locally by the $PF^{(r)}$. Thus we ensure that the product of local weights indeed satisfies (4.8) by performing the local computation:

$$w_k^{(r),i} \propto w_{k-1}^{(r),i}\, p(y_k^{(r)}|x_k^{(r),i}). \tag{4.14}$$

A coordination of all the $PF_r^{'s}$ remains necessary for normalisation, because

$$\sum_i \prod_r w^{(r),i} \neq 1$$

if one uses local normalisation $\sum_i w^{(r),i} = 1$ for all r. Resampling — for which we use the same strategy as the centralised PF, i.e. copying the local states of the most plausible particle group $\mathcal{P}_i$ defined in (4.13) — also needs a central station to consider the plausibility of a particle i over all r. But these two operations are not too time-consuming, requiring only one network-wide exchange of real numbers at the end of each iteration. To save computation time, one could even apply them only every few

iterations, with virtually no loss in estimation performance. Note that in the case of the distributed particle filter, $w_{k-1}^{(r),i}$ is computed from local measurements $y_{1:k-1}^{(r)}$ in region (r) only, but measurements taken in other regions will affect the particle set of (r) through the resampling action.

Distributed particle filter for urban traffic

The partitioning principle applies very well to the modular urban traffic network model: neighbouring components can share information about traffic flow at their boundaries through the variables $b_{k-1}^{(r)}$ which describe the (part of the) platoons that cross the boundary during the interval $[(k-1)\Delta_s,\, k\Delta_s)$.

Consider again the network $\aleph$ defined in Section 3.4, with N_{sens} sensors deployed along its links.

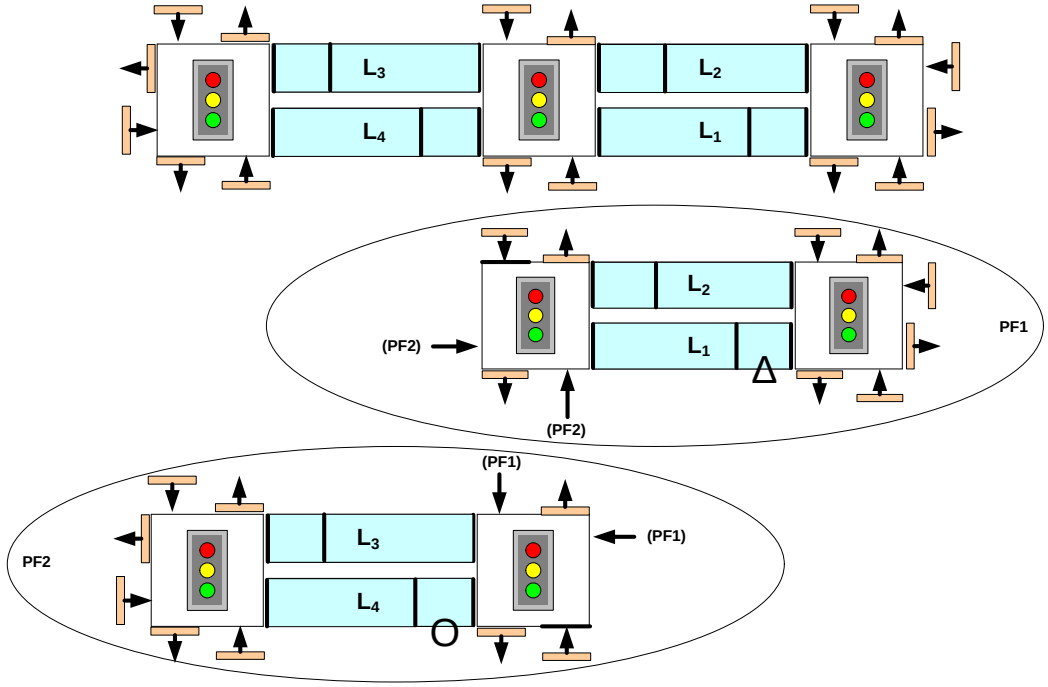


Figure 4.2: A small urban traffic network and its partitions for the DPF

We partition the network as illustrated in Figure 4.2: typically, each component

takes care of a few links, sources, sinks, inner intersections, and the selection lanes of intersections that serve as connecting components; the method is flexible enough to accommodate other partitioning schemes, e.g. cutting between two sections of a link. The $\mathcal{N}_R$ local particle filters PF_r associated to the partitions work in parallel by recursively applying the steps described by Algorithm 4.2.

The steps are essentially the same as for the centralised PF, with a few exceptions.

- Before each prediction step, PF_r sends the shared information $b^{(r*),i}_{k-1,r}$ (traffic flow at boundaries) to all its neighbours; in return it receives from them all the $b^{(r),i}_{k-1,*r}$ making up its own $b^{(r),i}_{k-1}$ needed for the prediction.

- A correct normalisation requires that all local weights are sent to some central station which computes and broadcasts the weight of each consistent particle group in $\mathcal{P}_i$ defined in (4.13) that represents a plausible overall network situation.

- After normalisation, $w^{(r),i}_k$ and w^i_k contain the same information so one can work indifferently with either of them for ordering/output and resampling. The resampling procedure adapts the local weights — no further communication required — with a modified formula (lines 21-22 in Algorithm 4.2) that ensures correct weighting of the particle groups $\mathcal{P}_i$ such that the products of the weights of each group $\mathcal{P}_i$ sum to 1.

The big advantage of the distributed approach is that the size of the traffic network does not represent an issue for simulation, since each PF_r runs on a local computer, with local information only (simulating the platoons present in the local area r). Some global coordination is still necessary for resampling, but note that the most heavy variables, that is local measurements and states, never need to be broadcast. Also remember that these global operations need not necessarily take place at every iteration, but only when the output is needed. More flexibility can be added by adapting the local number of particles to actual requirements, e.g. using $M \cdot \mathcal{N}_\mathcal{P}$ particles in a busy region to represent $M > 1$ possible local situations compatible with a single particle group i of the other regions. The DPF has the disadvantage of introducing a small approximation, because platoons that cross the boundary between partitions are signaled only at communication rounds, i.e. every Δ_s [sec], while the centralised PF takes them into account in the exact continuous-time agenda.

Algorithm 4.2 Particle filter PF_r of subnetwork r in a distributed particle filter for urban traffic

Ensure: $\rightarrow$ Initialisation of the particles at $k = 0$
1 **for** $i = 1$ to $\mathcal{N}_\mathcal{P}$ **do**
2 generate the i^{th} sample $\left\{x_0^{(r),i}\right\}$
3 initialise i^{th} weight with $w_0^{(r),i} = 1 \,/\, (\mathcal{N}_\mathcal{P})^{\mathcal{N}_R}$
4 **end for**
5 **for all** k such that $k \geq 1$ **do**
Ensure: $\rightarrow$ Send $b_{k-1,r}^{(r*),i}$ to all neighbours $r*$
Ensure: $\rightarrow$ Prediction step
6 **for** $i = 1$ to $\mathcal{N}_\mathcal{P}$ **do**
7 propagate i^{th} sample $\left\{x_k^{(r),i}\right\}$ with $p(x_k^{(r)}|x_{k-1}^{(r),i}, b_{k-1}^{(r),i})$
8 **end for**
Ensure: $\rightarrow$ Update step
9 **for** $i = 1$ to $\mathcal{N}_\mathcal{P}$ **do**
10 update the weight of the i^{th} sample according to $w_k^{(r),i} = w_{k-1}^{(r),i} \cdot p\left(y_k^{(r)}|x_k^{(r),i}\right)$
11 **end for**
Ensure: $\rightarrow$ Send $\{w_k^{(r),i} : i = 1, 2, ..., \mathcal{N}_\mathcal{P}\}$ to central station
Ensure: $\rightarrow$ Normalisation
12 get global weights $\{w_k^i = \prod_{r=1}^{\mathcal{N}_R} w_k^{(r),i} : i = 1, 2, ..., \mathcal{N}_\mathcal{P}\}$ from central station
13 compute new total weight $W_k = \sum_{i=1}^{\mathcal{N}_\mathcal{P}} w_k^i$
14 **for** $i = 1$ to $\mathcal{N}_\mathcal{P}$ **do**
15 replace $w_k^{(r),i}$ by $\sqrt[\mathcal{N}_R]{w_k^i / W_k}$
16 **end for**
Ensure: $\rightarrow$ Output
17 order particles by their weight.
18 $\hat{X}_\aleph^r \leftarrow$ the state of the particle(s) with the highest weight(s).
Ensure: $\rightarrow$ Resampling
19 **for** $i = 1$ to $\mathcal{N}_\mathcal{P}/2$ **do**
20 replace the state of the particle $x_k^{(r),N_P-i+1}$ with the state of the particle $x_k^{(r),i}$
21 $w_k^{i_{new}} = \left((w_k^{(r),i})^{\mathcal{N}_R} + (w_k^{(r),N_P-i+1})^{\mathcal{N}_R}\right) /2$
22 $w_k^{(r),i} = \sqrt[\mathcal{N}_R]{w_k^{i_{new}}}$ and $w_k^{(r),N_P-i+1} = \sqrt[\mathcal{N}_R]{w_k^{i_{new}}}$
23 **end for**
Ensure: $\rightarrow$ Time step update
24 $k \leftarrow k + \Delta_s$
25 **end for**

4.4 Validation of the PF estimators

Available data sets of real traffic give only parameters like inflows, outflows and turning ratios at different locations where sensors are installed, but no accurate information on queue sizes nor *a fortiori* on instantaneous locations of platoons. We therefore validate the estimation algorithms on synthetic data, produced by simulation of a *ground truth model* (GTM). The latter generates a trajectory for the whole state (including the size of queues and locations of all vehicles), but only observations taken every $\Delta_s = 3[\text{sec}]$ are transmitted to the PF.

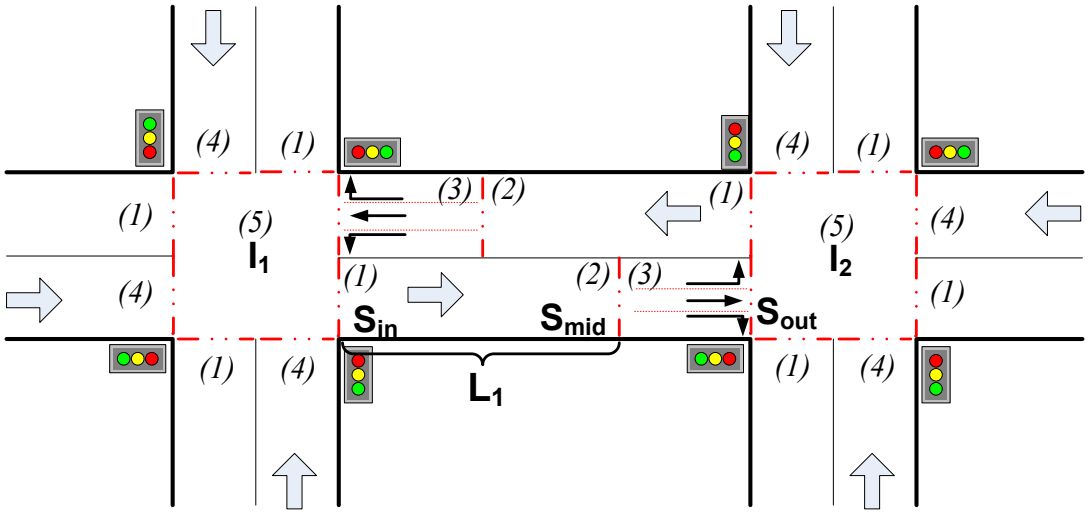


Figure 4.3: The road topology and the sensor's locations for the validation study

We first use the PBM itself as GTM, to check correctness of the PF algorithm alone. Then to validate the PBM, we generate synthetic traffic data with the well established micro-simulator SUMO (Simulation for Urban MObility, see [3, 44]). We have added noise to the perfect measurement outputs generated by SUMO in order to match the data quality that our sensors seemed to give in the Dendermonde data set. For the simulations, the parameters of the model for the PF prediction step are taken over from the GTM; in a real-world situation, they would be estimated from historical data. An advanced version of the algorithm could consider adaptive parameter estimation where

model parameters (e.g. turning ration, distribution of platoon sizes at sources, sensor noise levels) are estimated on-line and adjusted accordingly in the PF.

Figures 4.4, 4.5 and 4.6 show the queue sizes generated by the GTM and those estimated by the particle with the highest weight. At each given instant for the entrance points of intersections, we show the sum of the queues on all preselection lanes. For easier interpretation, green blocks at the bottom indicate when the traffic light is red for the section for which the queue size is shown. Thanks to our particular resampling strategy, it indeed makes reasonable sense to approximate the most likely estimated state by the most weighted particle, although this is not strictly correct. Like every filter, the PF in fact gives as estimation a "belief histogram", i.e. a full probability distribution of queue-sizes associated to the model and measurements at each time. The reason for representing only the most likely particle is that computing and analysing this histogram would not be feasible for real-time control purposes. The $\mathcal{N}_\mathfrak{P}$ values are chosen to illustrate realistic possibilities: to be able to run the PF in real-time, the computation time for running $\mathcal{N}_\mathfrak{P}$ particles over an interval of length Δ_s must be smaller than Δ_s. More insights about using the histogram for estimating the state are given in Subsection 4.4.4.

4.4.1 Centralised PF with the PBM as ground truth model

Figure 4.3 shows the network used for this validation, using sensors located at S_{in}, S_{mid} and S_{out}. The first example shows how the PF can cope with an accident that suddenly (at $t = 360[\text{sec}]$ on Figure 4.4) decreases the speed V_{max} of platoons that enter a link (from $60[\text{km/h}]$ to $25[\text{km/h}]$ on Figure 4.4). The PF allows for an "accident" possibility by letting a random number of particles embed a random drop in speed at each Δ_s step. We assume that an accident can take place with a probability of 10^{-5} in reality and with 10^{-2} in particles. A 10^{-5} probability for an accident to happen would mean 10^5 particles such that at least one includes the accident. Thus, we include a higher probability of an accident to happen in the PF. The algorithm adapts the weights of those "accident" particles according to their likelihood, given the probabilistic model and measurements (this requires *importance sampling* to deal with the rare event of an accident).

Figure 4.4 shows the evolution of the estimated queue size (pink dashed) and of

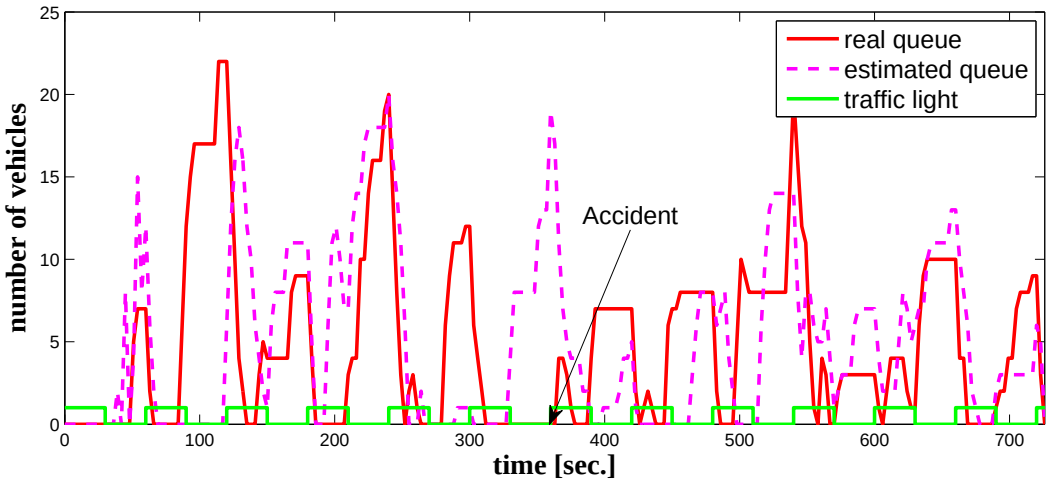


Figure 4.4: Estimated queue (pink dashed line) vs. real queue (full red line) generated by the PBM, in presence of a sudden accident

the real queue (full red). The traffic lights switch every $30[\text{sec}]$. The PF follows the real queue most of the time until $\approx 270[\text{sec}]$, where the estimator sees many vehicles arrive up to $50[\text{sec}]$ later than they really do; this might be due to occasional abnormally bad sensor measurements. Just after the accident, the PF is under-estimating the real queue, until it recovers after $\approx 100[\text{sec}]$. The computation time for this example was $2.2[\text{min}]$ to analyse $12.1[\text{min}]$ of real time using 202 particles, with a naive MATLAB implementation. The accident is not investigated anymore in the following examples.

4.4.2 Centralised PF with SUMO as ground truth model

The previous example could be considered too optimistic since the ground truth model and the PF model are the same. Therefore we let SUMO to select the parameters for traffic lights (switching every $45[\text{sec}]$) and route characteristics for the same network of Figure 4.3; model parameters (e.g. vehicle length $5[\text{m}]$, $v_{max} = 50[\text{km/h}]$,...) are taken over from SUMO in the PBM implemented by the PF, assuming that in practice they can be reliably estimated.

Figure 4.5 shows how the PF estimate follows the real queue. The different ground-truth model leads to a slower convergence time and mostly an overestimation of the real

queue. This result is obtained with 300 particles, with a computation of $2.7[\min]$ for $20.05[\min]$ of traffic time.

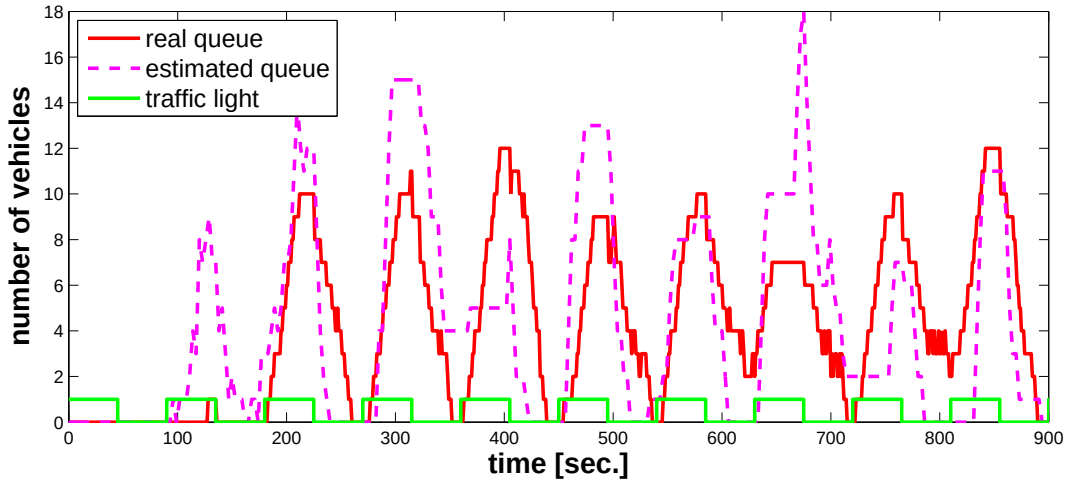


Figure 4.5: Queue estimated by the PBM-based central PF (pink dashed line) vs. real queue generated by SUMO (full red line)

The algorithm runs faster (computation time vs. real time ratio) in the case of the second example compared with the first example. Even though, we use a larger number of particles in the second example (as summarised in Table 4.1).This is due to the fact that in the second example we do not use the importance sampling for accident detection.

	Computation	Real time	N_p	ratio
Section 4.4.1	2.2	12.1	202	0.18
Section 4.4.2	2.7	20.05	300	0.13

Table 4.1: Computation time vs. real time.

4.4.3 Distributed PF with SUMO as ground truth model

Since the proposed DPF algorithm is fully modular, our validation only needs to consider interaction between two subsystems. We therefore consider the network with 3

intersections shown in Figure 4.2. The state vector of the GTM is in this case partitioned in two subvectors. The middle intersection connects $PF1$ and $PF2$. The same SUMO simulator is used to generate synthetic data with the network of Figure 4.2 and traffic lights switching every $40[\text{sec}]$). The prediction errors of the queue sizes are presented in Figure 4.6, using 500 particles in the centralised PF, and 300 particles for each subsystem in the DPF. The diversity of particles in the centralised PF was observed to decrease faster than in the case of the DPF, therefore a higher number of particles is necessary in the former approach. The running time for the DPF is similar to the previous cases but, since the CPF and the DPF run on the same computer, a quantitative analysis cannot be performed. As expected, the distributed and centralised versions both appear to estimate comparably well. The jumping behaviour is typical of representing a whole probability distribution by just one most likely particle: in case the distribution is flat, very small changes in weights can lead to selecting very different particles. Convergence is similar to the previous case, often giving a slight overestimation of the real queue. A meaningful metric to exactly quantify how well the respective estimators serve their purpose, would be how well each one allows to control the traffic system; the controller development is subject of the next section of thesis.

4.4.4 Histograms of the distributed PF with SUMO as ground truth model

We already mentioned that the particle filter returns at each iteration a full probability distribution of queue-sizes based on the model and the available measurements at each iteration. Figure 4.7 presents the histograms of the centralised PF and of the distributed PF (PF1) compared with the real queue size (the plotted red line at the top of the histograms) at each time instant for one of the links. The most likely estimated state is approximated by the most weighted particle at each given instant. The drawback is that for flat belief distributions (high uncertainty in the PF with many weights almost equal to the maximum weight), selecting one most likely value leads to high variability from one time instant to another time instant explaining the jumpy estimation results (for example see around time 170 on Figure 4.7). For better robustness, controllers could also consider a simple uncertainty estimator, like e.g. the weighted value of the selected particle, or the similarity between the 5 or 10 most likely particles. Such indicators

require evaluation in combination with the controller, so they are left for future work.

To represent the probability distribution properly with samples, we want the weighted density of the particles in any subset of the state space to be proportional to the probability of that subset. Thus, the weighted density of the samples in one area of the state space represents the probability of that region. For a best-guess of the most likely estimated state, we can use the histogram with weights to obtain the region with the highest weighted density of particles, or the weighted sum of particles to get a mean-equivalent.

As can be observed in Figure 4.7, most of the time only one value has a weight significantly different than zero. Based on this observation, we could show just the value of that peak. Given our weight-conserving resampling mechanism, we hope – again without any formal guarantee – that the particle with highest weight value will correctly identify the histogram peak.

An information useful to present would be platoon location estimates, not only queue sizes. For example one particle that has five vehicles in a queue and other particle which has only two vehicles in a queue but three other vehicles just a short time away from joining/leaving that queue, represent in practice the same traffic state. In fact, this variation is very likely to appear since in reality vehicle speed is known to vary, while our model when grouping vehicles into a platoon makes the approximation that all the vehicles have the same speed. Therefore, a better way to evaluate the likelihood of particles would be to compare the deviations in time of expected vehicle detections (according to the particle) and actual sensor signals. But presenting platoon location estimates would have several implications:

- the visualisation of the platoon location estimates is not straightforward to be made in a way that is useful for control applications,
- more importantly, in presence of such hybrid uncertainties in motion dynamics and in sensors, a more advanced procedure should be investigated for sharing the uncertainty between generating particles and evaluating their likelihood: the weight should not just include sensor uncertainties, but also account for some of the modelling uncertainty in order to lower the burden on the particle sampling.

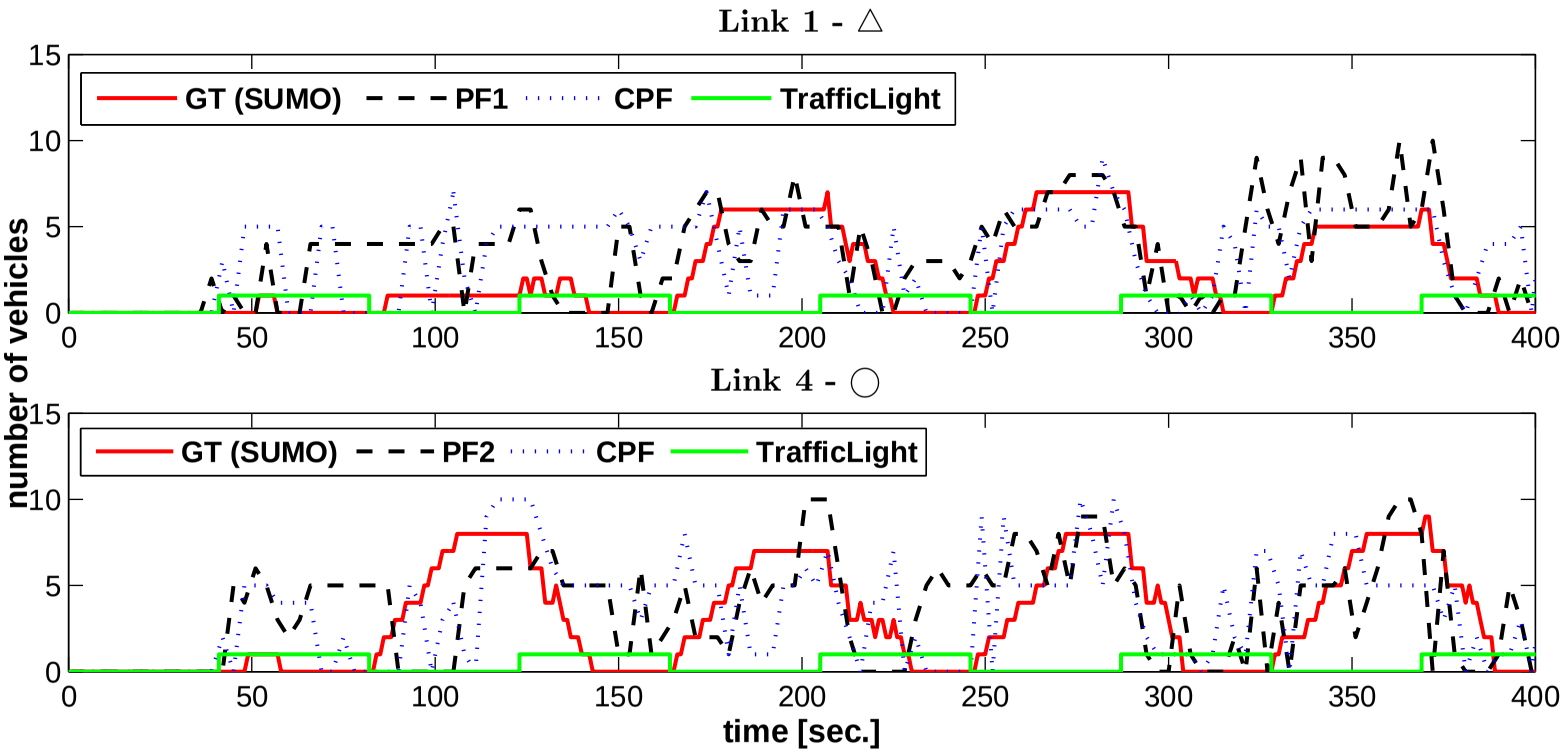


Figure 4.6: Centralised PF (CPF) and distributed PF (PF1 and PF2) queue estimates vs. ground truth, for two labeled links in the network of Figure 4.2.

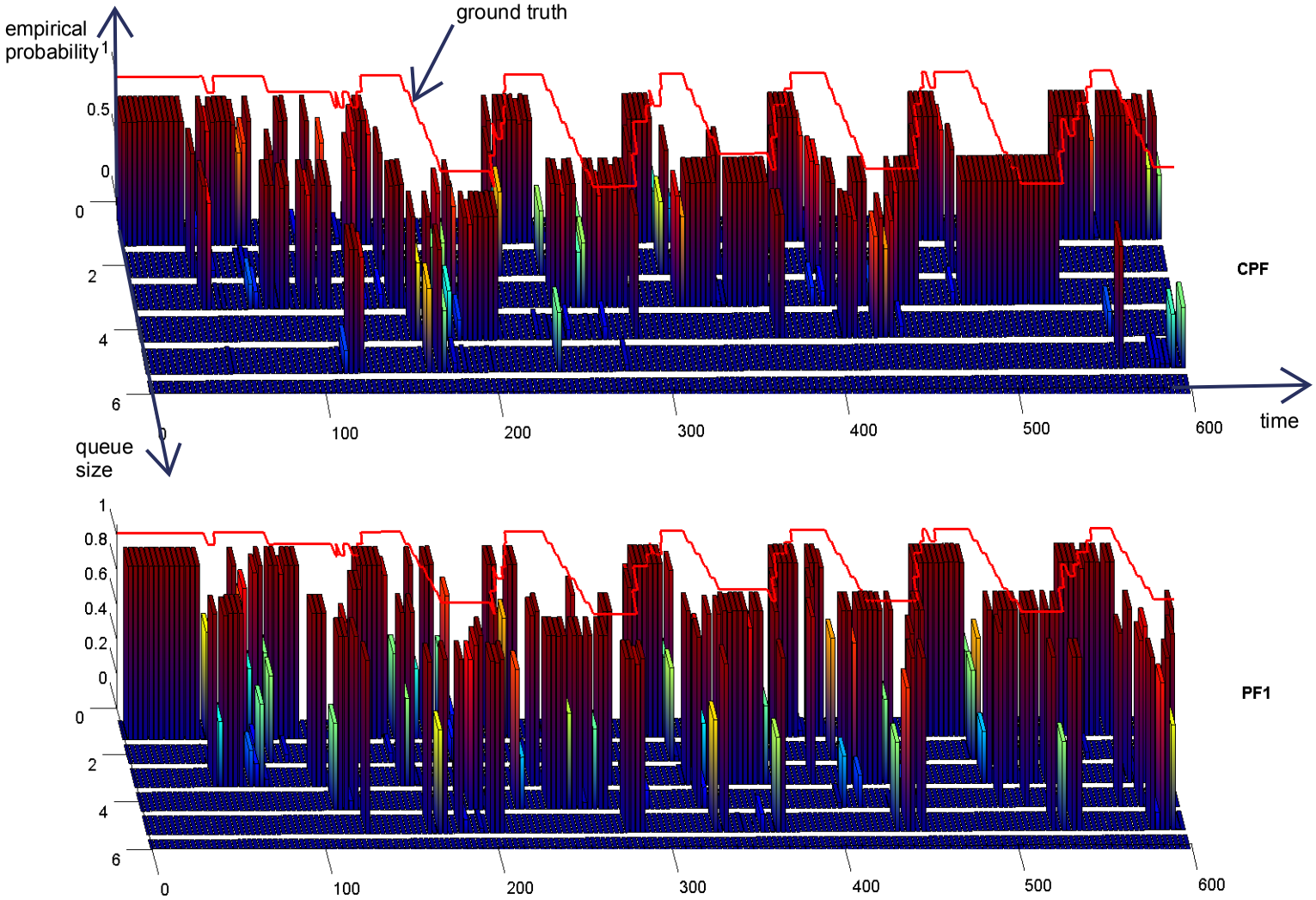


Figure 4.7: Centralised PF (CPF) and distributed PF (PF1) histogram vs. ground truth (presented in red line) , for $Link_1$ labeled in the network of Figure 4.2.

4.5 Summary

In the current chapter we used the platoon based model, introduced Chapter 3, for generating particles for a particle filter estimator of the state of an urban traffic network. First, estimation issues are discussed using again the analysis of the raw traffic measurements followed by an argumentation for the use of the particle filter. After the theory behind the particle filter was reviewed, a centralised particle filter algorithm is proposed. Since a centralised solution is not feasible for large networks a distributed particle filter is developed. The platoon based model and the two algorithms are validated using synthetic data produced by a third party model.

The obtained results show that the particle(s) with the highest weight(s) convey reasonable information on true system behaviour. Thus, we expect the output of our estimators to be useful for control.

Chapter 5

Distributed feedback control of urban traffic networks with leader-follower coordination

The congestion of urban traffic networks is a reality of our days. Possible solutions to ease the congestion may consist in expanding the current infrastructure, in developing and encouraging public transportation or in improving the network performance through a more efficient use of the existing infrastructure. The performance of the network can be improved by controlling and coordinating the traffic lights of the network.

In this chapter, a new distributed feedback control framework is developed for the control of urban traffic networks under intermediate load for two-way traffic. Local control agents use model based feedback controllers to select the switching times of traffic lights of the network. The coordination between the agents is achieved by using the leader-follower paradigm, as follower agents take into account the implications of their switching choices on the timing of traffic sent to neighbouring leader agents. The aim is to simultaneously minimise the waiting time of all vehicles by combining a real-time reaction to traffic arriving at local intersections with a green wave type objective, where platoons of vehicles would travel through consecutive intersections without having to stop.

We start by giving a short survey on control of urban traffic networks in Section. 5.1 and show how it compares with our work. Section 5.2 contains a preliminary control

analysis of different traffic loads. The proposed control strategy is introduced in Section 5.3 followed by various validation examples presented in Section 5.4. The chapter ends with a robustness evaluation shown in Section 5.5.

5.1 Control of traffic

An urban traffic network $\aleph$ consists of roads connecting signalised intersections controlled by traffic lights (see Definition 3.4). Traffic lights become inefficient due to waste of capacity (e.g. green without traffic, yellow) and the control of the traffic lights should minimise this wasted capacity. Compared to our work, where we use the basic control variable "switching time", the large majority of other approaches uses the *cycle length*, *green split* and *offset* as traffic light parameters. A phase corresponds to a particular state of the traffic light (e.g. green, yellow, or red). The *cycle length* represents the amount of time that it takes for a traffic light to make a complete cycle of phases. A long cycle length may cause a large waiting time for a stopped platoon of vehicles while a short cycle length leads to shorter waiting times with frequent stops, but also wasting yellow time. The relative duration of the green phases for different directions of the intersection is called the *green split*. The last parameter, the *offset*, is the phase shift between cycles of neighbouring traffic lights. By properly adjusting the offset, one can create a green wave. A green wave is the induced phenomenon that creates a progressive cascade of green over several neighbouring intersections. Useful effects of a green wave are to create and maintain platoons of vehicles that do not have to stop very often at intersections, reducing the noise (produced by vehicles accelerating or decelerating which usually occurs near junctions or at intersections) and the fuel consumption.

Depending on how the parameters of a traffic light are determined, we can distinguish:

- fixed time switching plans (FTSP) - lead to static switching plans for a given time-of-day. The switching plans are determined using off-line optimisation solutions based on historical parameters of the traffic demand and long term predictions (e.g. several hours). The real-time dynamic aspects of traffic are not captured by this type of solutions. Event though the FTSP can be updated, instantaneous

changes of the static switching plans may lead to transient effects that destroy the green waves. See [20, 78, 79] for more details about fixed time switching plans.

- traffic-responsive switching plans (TRSP) - use real-time measurements and short term predictions (e.g. minutes) to calculate in real-time the parameters suitable for setting the traffic light according to the traffic demand. The recalculation of the parameters depends on how fast the system reacts to the traffic changes and may cover different time horizons (e.g. one cycle to several cycles). These adaptive control approaches need sensors on the roads, local controllers in each intersection, communication networks and/or traffic control centres. Depending on the control architecture, one can distinguish:
  - isolated control solution - applicable to a single intersection. If applied to a network of intersections it leads to fully distributed solutions with the ability to locally adjust the cycle length and green split of each intersection, but without any communication or coordination with neighbouring intersections. The local decisions make the creation of green waves impossible. The network does not share a common cycle length and has cycles with arbitrary offsets. A common cycle length and offset demand a periodicity that is restrictive from the dynamic perspective of traffic. The following items use all three parameters (*cycle length*, *green split* and *offset*). This kind of solution is presented for example in [42];
  - centralised solutions - applicable to the entire network. This type of solutions are network-solutions without the possibility of adding intersections to the network in a ”plug-and-play fashion”. The solutions use for control all three parameters (*cycle length*, *green split* and *offset*). The computational burden is high since the optimisation is done for the entire network at once and depends on the communication network; (on-line, real-time) timing of signal phases at traffic signals, meaning that it tries to find the best phasing (i.e. cycle times, phase splits and offsets) for the current traffic situation (for individual intersections as well as for the whole network). This kind of solutions are for example presented in [14, 37, 38];
  - coordinated control - in contrast to centralised, these solutions are decentralised with local communicating agents that try to find the best control

actions (i.e. *cycle length*, *green split* and *offset*) in real-time, for individual intersections and for the entire network. Such solutions consider a region with a few intersections or even an entire network comprising many intersections. This kind of solutions can be hierarchical control solutions (parameters are determined at different layers - green split by a low layer and cycle length and offset at a higher centralised layer) and work in a "plug-and-play fashion". This kind of solutions are for example presented in [31, 49, 61, 81].

We propose a coordinated control solution with two layers: a low layer of local controllers (also called agents) deployed in each intersection, and a supervisory higher layer. However, we allow the local controllers to communicate with neighbouring intersections and devise a leader-follower paradigm that implements coordination already at this lower level. This should minimise the necessity of interventions by the supervisory layer, which can be crucially important for scalability to very large networks. Traffic is described by specifying the arrival times and size of platoons of vehicles at sensor locations in the network. The important aspects of our approach are:

- An agent uses a model based feedback controller to locally optimise its next switching times with two goals to minimise local waiting queues and to coordinate with its neighbours. We need a model like the PBM, introduced in Chapter 3, to provide enough details and to compute very fast the effects of many different possible scenarios for the next switching times. Using the PBM, we can predict the arrival times of the platoons which already entered on the roads upstream of the intersection until time δ_l (the time that it takes to drive the entire length of road l at the maximum legal speed for the same road). After time δ_l, the traffic flow coming from the upstream intersection can only be replaced with average platoon arrivals unless information about the state of the upstream intersection would be available. The optimisation that results in selecting the next switching times is carried out over a time horizon (not too long into the future since after the time δ_l the real traffic is averaged, but sufficiently long to see the coordination effects). Since there is no real traffic data for the second switching, and also for computational reasons, the current analysis looks at one next switching time. Undoubtedly, if traffic data in the network is available to predict the detailed

traffic evolution at a given intersection further into the future, the method can be extended to looking at several switching times ahead with further improvements (e.g. in programming the simulation tool) and better insight into the choice of parameters (e.g the time used to recompute the next switching times, rejecting many uninteresting cases of switching times from the start);

- Coordinating the interacting local controllers of the urban network requires to share control intentions between agents. The coordination goal is to make the local controllers help each other to achieve the global goal of creating dynamic green waves with respect to traffic changes. For achieving coordination in case of intermediate traffic load (when only a few critical intersections are heavily loaded), the agents of these critical intersections are selected as leaders. The neighbouring intersections of the leaders can then be selected as follower agents. The leader agents send specifications - the next optimal switching times - to their follower agents. A follower agent incorporates the specifications of the leader in its next switching times or it pays a deviation price from the specifications of the leader. By following the leader's specifications, it makes the leader intersections minimise their waste of capacity (e.g. minimise the fraction of time when green light is given to a direction where there is currently no queue nor any arriving platoon), while ensuring that the follower intersections do not become overloaded. This coordination strategy will reduce the risk that the network becomes overloaded (e.g. gridlocked in the case of urban traffic);

- The supervisory layer makes a time dependent leader-follower assignment determined via feedback using sensor data on the current traffic load distribution. Thus, the supervisor detects which intersections are critical, on the basis of some origin/destination flow rates (estimated from aggregated measurements received from all sensors in the network). The re-assignments of leaders should be very rare events compared to the time scales of switching traffic lights. We leave the development of the leader-follower assignment as future work.

Focused on the approach proposed by us, we looked at other solutions close to different aspects of our work (e.g. similar control architecture, similar traffic models, etc.).

5.1.1 Literature context

The aim of this section is to present those adaptive control methodologies currently in wide use that have some theoretical or fundamental similarity to our work. Major solutions, widely used in the field and real-world implementation are OPAC-[31], SCOOT-[37] , SCATS-[49], RHODES-[61]. Each solution is explained briefly and its important aspects are highlighted.

Optimised Policies for Adaptive Control (OPAC - [31]) is a solution first proposed in the early '80s in a study for the US Federal Highway Administration. It is a real-time hierarchical control solution that can work either as a distributed control system or as part of a coordinated system. A low layer controller manages each individual intersection independently and computes optimal switching sequences for the projection horizon, subject to a variable cycle length within bounds as a constraint. A flow profile is developed for each phase of the intersection using a designer specified control horizon length, where the head of flow profile uses observed counts of actual traffic from the upstream link detectors and the tail of profile is projected for the near future using smoothed volume counts. Based on the platoon identification, the cyclic profiles are determined. At this level, a decision is made whether to terminate the current phase or extend it by one interval (1 or 2 seconds). An intermediate layer performs a real-time optimisation of offsets at each intersection. The decision on a new offset is made at the end of the current cycle. A high layer performs a network wide signal synchronisation.

Split Cycle Offset Optimisation Technique (SCOOT-[37]) is a real-time centralised solution developed in the early '80s by the Transport Research Laboratory in the United Kingdom. The solution is based on once-per-second mandatory communication with each local controller. None of the optimisation steps is performed in the local controller. SCOOT measures vehicles at a detector ideally placed at the upstream end of the links. Every second, a central program makes predictions (for each link) of the arrival profiles and queues based on the measurements at each detector. For each link, SCOOT creates traffic profiles as a combination of flow and occupancy called link profile units (one vehicle is equivalent to approximately 17 units, but the actual value is different for each link). This is a commercial solution and the computation of these units is not explicitly mentioned in SCOOT. Based on these profiles, it performs three optimisations using as control variables the split, the offset, and the cycle as follows:

- A few seconds (e.g. 5 seconds) before switching the traffic light at every SCOOT intersection, the *split optimiser* estimates whether it is better to make the change earlier or later with the objective to minimise the maximum degree of saturation;

- At each intersection, the *offset optimiser* makes the offset decision once every cycle based on the flow profile information and estimates of overall traffic progression. Since the offset of one intersection is altered relative to neighbour intersection, the offset between an adjacent pair of intersections may alter twice per cycle;

- After the controlled intersections are grouped into sub areas with pre-set boundaries, the *cycle optimiser* tries to find, for each subarea, a value for the common cycle length between preset upper and lower bounds. The lower bound is determined by the considerations of safety, pedestrian crossing time and minimum green time (e.g. 30 - 40 seconds). The upper bound is set to give maximum traffic capacity, without unduly long red times (e.g. 90 - 120 seconds). The cycle length is incremented or decremented to ensure that the most heavily loaded intersection operates at an optimally balanced degree of saturation (the ratio of demand to capacity) on each approach to the intersection.

The optimisers make frequent changes by small alternations to the parameters in order to adapt the fixed time plan to variations in the traffic behaviour. As a disadvantage of the centralisation, for operation, SCOOT entirely relies on the communications network and central computer.

Sydney Coordinated Adaptive Traffic System (SCATS-[49]) is a real-time hierarchical solution created in the early '90s by the Roads and Traffic Authority of New South Wales, Australia. The system operates by looking at space between vehicles (space time is inversely proportional with the traffic density) and the saturation degree of each intersection. The architecture of SCATS has three layers as follows:

- A centralised computer used for monitoring the overall system performance and the status of regional computers;

- A regional computer autonomously controls the local controllers of a region. A form of aggregation is used at this level, where local controllers within a region

with the same homogeneous flow characteristics form subsystems (up to ten intersections). Several subsystems together form a system. The subsystems within a system are dynamically changing with the traffic load (e.g. an intersections can be reassigned to a region with a similar flow characteristics). Using a strategic control algorithm based on the time-of-day and average traffic information, it selects a scenario containing the cycle length, green split, and offset on a cycle-by-cycle basis. The regional computer selects from a library of offsets and phase splits to optimise timing plans in real-time. All the intersections from a subsystem operate on a common cycle length determined by the intersection with the highest saturation degree during the previous cycle.

- Local controllers, one for each intersection, pass average traffic information to the regional computer and receive the current selected scenario. At this level, each local controller makes the ”fine tuning” of the current phase (terminated if the demand is lower than predicted, entirely omitted, or maximally extended).

The adaptivity of SCATS is based on the local actual controllers without using a traffic model. Missing a traffic model limits the use of an optimisation approach. Another drawback of the model-free approach of SCATS is not using the anticipation of arriving traffic.

Real-time Hierarchical, Optimised, Distributed, Effective System (RHODES-[61]) is a real-time hierarchical solution produced in the mid-’90s at the University of Arizona at Tucson. Entirely based on dynamic programming and using vehicle arrival data, RHODES formulates a strategy that makes phase-switching decisions. The optimisation is performed at 3 different layers as follows:

- The macroscopic layer estimates changes in the aggregated flow of data over the entire network. The estimates (link flows as vehicles per hour) are transmitted to the middle layer;

- The middle layer uses a mesoscopic model based on the identification of platoons and average speeds. The network is decomposed in sectors (e.g. arterial). Every 200 to 300 seconds, the green phases (cycle length, green split, offset) and their order (assuming that there are more than two green phases per cycle) are calculated to minimise the delay of platoon movements using simulation, and a

decision tree method such that the coordination between intersections is achieved. The parameters of the green phases are sent to the lower layer;

- The lowest layer is represented by the local controllers present in each intersection. The controllers use a microscopic model and perform a second-by-second fine-tuning of the green times and phase ordering sent by the middle layer. The vehicle arrival estimates at the intersection and the network flow information from the upper layers determines whether the current phase should be extended or not.

In conclusion, SCATS is more coordinated compared with SCOOT but does not use the anticipation of arriving traffic. The coordination is purely made by exchanging messages on switching times of traffic lights. SCOOT on the other hand does not have coordination through message exchange, but it has anticipation by sensing the traffic at the upstream end of a link (i.e. at the exit of the upstream intersection). Thus, SCOOT achieves coordination more by anticipation.

Our method in conjunction with the platoon based model, combines the two methods of coordination present in SCOOT and SCATS. Even further, the functionalities of the lower layers of OPAC and RHODES are performed by a single layer in our case with the optimisation rerun after a time period less than a cycle (e.g. 1 second in our experiments). In our case, only minimum and maximum phase lengths are specified (the cycle length control variable is not used by our algorithm and depends solely on traffic load), which gives an even higher flexibility for reacting in real-time to the changes. In [75], the practice of using a common cycle length is under question. Using the high cycle length of a heavily loaded intersection common for a region of intersections affects the waiting time for the traffic in lightly loaded intersections.

Starting from a mathematical model of traffic, some papers try to develop a control strategy for a single oversaturated intersection and extend it to networks. In [46], using a discrete event model, the problem is decomposed with respect to time to handle time delays into simpler linear quadratic problems. The idea is extended in [22], where optimal switching schemes are designed for linear dynamics switched systems by the Extended Linear Complementarity Problem (ELCP) using a fluid flow model. Both consider as input data the arrival and departure rates of the vehicles at the intersection and as state the queue lengths. In the case of [46], the green split is used as control variable with a fixed cycle length, whereas [22] is a cycle-by-cycle solution that allows

the cycle length to vary from one cycle to another, and several possible objective functions are investigated. Similarly, we first develop the platoon based model and based on it, a novel framework for the distributed feedback control of urban traffic networks with leader-follower coordination. Additional results can be found in [36, 47, 48].

5.1.2 Platoons in control

Further, we look at different papers where platoons of vehicles are proposed to be used in the control of traffic lights. Although the literature on platoon based models is not very rich, the existing papers show that this type of traffic representation is well suited for control applications.

In [42], an algorithm is developed to reduce traffic delays at a single major-minor intersection by minimizing the interruptions of the platoons from the major road. The waiting time for vehicles on the minor road is limited to a maximum waiting time as a tradeoff for minimizing the traffic delays and giving priority for the major road. An interesting analysis of the characteristics of platoon-based traffic flows is made using real measurements.

A centralised solution to synchronise traffic signals along an urban artery through a platoon based simulation model is proposed in [14]. The traffic model provides the queue length, the average delay at intersections, the departure time and the time length of platoons travelling along the urban artery. A genetic algorithm combined with a hill climbing algorithm are developed. The objective function is a linear combination of the total delay on each direction of the artery and the total delay at the approaches of the intersections. After a preset number of steps, the genetic algorithm stops and the hill climbing algorithm tries to improve the objective function through local adjustments based on a set of predetermined steps. The adjustments are first applied incrementally followed by symmetrically decreasing steps for each control parameter (cycle length, green split, offset) of each intersection.

A platoon-based self-scheduling algorithm is proposed in [81] aiming to maintain the movement of the platoons (i.e. to minimise the total delay of vehicles travelling through the network) instead of minimizing the queues. As a fully decentralised solution, each local controller has a limited view of the incoming traffic, and using a platoon-based heuristic approach, determines whether to extend or terminate the current

signal phase by a time increment given by the departure time of the platoons. The incoming traffic is represented for each link by the queue lengths and all the platoons (platoon size, the expected arrival time of the first vehicle and the departure time of the last vehicle from the platoon) that entered on the links. Two policies are proposed. A *platoon based extension* checks if there is a platoon on the road currently being serviced and decides whether to extend or not the current green phase. The second one, a *platoon based squeezing* checks if there is a platoon on the road that possibly arrives during red. The algorithm determines whether the current phase can be extended to best serve that platoon. Although the work is interesting, the examples use only "one-way traffic" which makes the synchronisation simpler, and no associated cost is used to make possible an evaluation of the tradeoffs made (i.e. to extend or not the current green phase). An interesting remark made by the authors is "... platoons can be used in a look-ahead horizon as patterns for facilitating implicit coordination". The remark is important since our aim is to use the platoon based model to reduce the computational complexity of the coordination problem.

Another platoon-based centralised solution for arterial multi-modal signal control is proposed in [38], assuming that the intended path which each vehicle will follow through the network is known to the controller. The algorithm first identifies the existing queues and a significant majority of the vehicles in the network. After this first step, a mixed-integer linear program (MILP) is solved to determine future optimal signal plans based on the current state of the controller, platoon data and priority requests from special vehicles (e.g. buses). The cost function includes the total weighted delay of platoons in the network and the sum of total remaining green time. The coordination algorithm is not based on a common cycle. Thus, the problem has a futuristic solution by taking in consideration the communication between vehicles and traffic controllers. Before introducing our new approach, we make a preliminary control analysis in the next section.

5.2 Preliminary control analysis

The goal of the current section is to compare control strategies for a single intersection in saturated, intermediate and very light traffic intensities assuming the full local state known. Based on this analysis, we extract conditions and make remarks that will help us develop our distributed feedback control framework for intermediate traffic conditions.

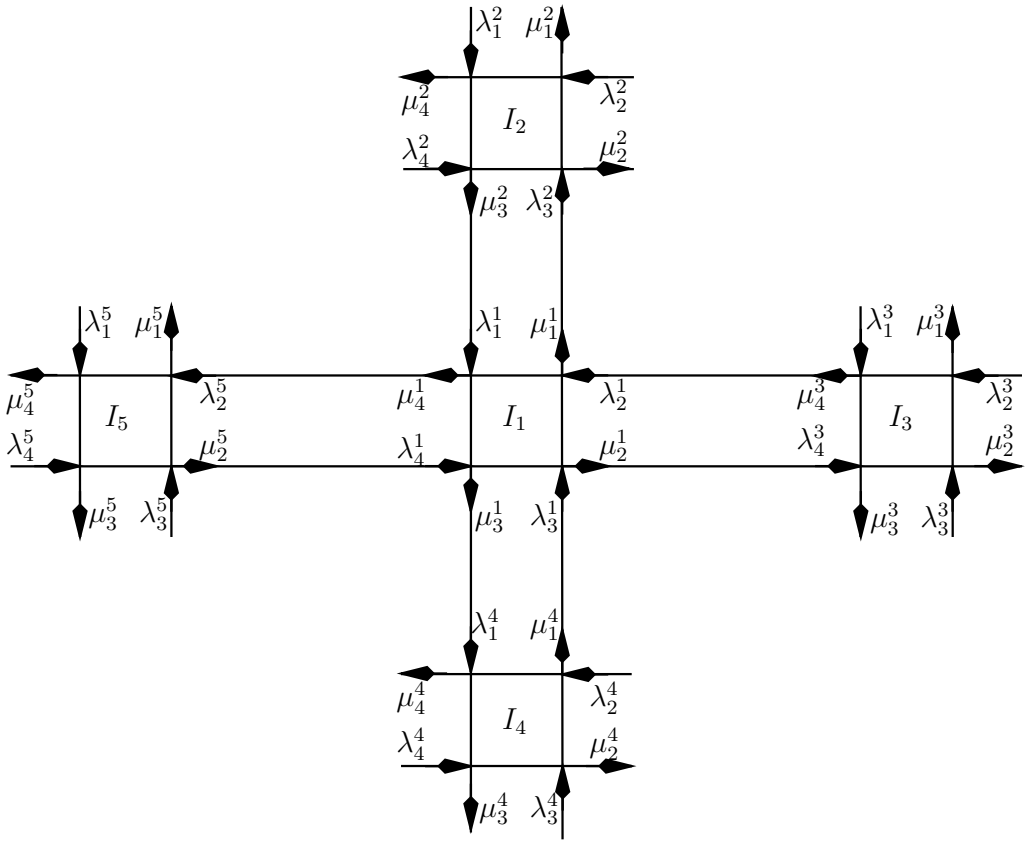


Figure 5.1: Five intersections interconnected in a star topology.

Considerations on the level of details for traffic models are presented in [68] as a criterion for selecting a suitable model for control. A model should have sufficient descriptive power of reproducing all the important phenomena with a simulation speed fast enough for the application that was intended. In [40], it is shown how an abstract (macroscopic) representation of traffic may provide better results for some applications than using a higher level of detail (e.g. microscopic models). The level of detail is

directly correlated with the model complexity and the computational load. The opposite, less details will make the model lose its applicability. Based on these considerations, we will use our platoon based model for the actual controller. For this preliminary theoretical analysis, we need even a simpler model and we use a fluid flow representation as presented in Section 3.1.1. An example of five intersections interconnected in a star topology forming an urban traffic network is presented in Figure 5.1.

5.2.1 A default switching rule for one saturated intersection

Before starting, we make the following remarks:

- λ_i^j is the arrival flow at input i of intersection j and μ_i^j is the maximal possible departure flow if queues are non-empty and traffic light is green at exit i of intersection j,
- the departure flow μ_i^j is the same for all possible exits i of intersection j and denoted as μ_j,
- λ_i^j and μ_j can vary with time; the analysis made in this section should be understood for values of μ_j and λ_i^j at a given time instant,
- μ_j is the instantaneously reached maximum departure rate (zero inertia assumption from Remark 3.1),
- the yellow period is assumed here to be zero, thus there is no waste of capacity for it (of course there may still be waste of capacity by giving green when no vehicle is present).

The last two items from the list imply what is called "stop-and-go traffic".

The control strategy $u_j(t)$ selecting the next $\mathcal{N}_c$ switching times at time t at intersection j is the set $u_j(t) = \{G_{j,c}^d,\ c = 0, 1, \ldots, \mathcal{N}_c - 1 \text{ and } d = 0, 1\}$ where $G_{j,c}^d$ is the c-th interval of green phase for the pair of directions characterised by d. We will sometimes make the abuse of notation that $G_{j,c}^d(t)$ is the indicator function of the respective interval, as in (5.1). The signal $\sum_c G_{j,c}^{d(i)}(t)$ then takes the value 1 for all t where direction i has green, and 0 for all other t.

The evolution of all the queues formed in the front of the traffic light in the intersection j can be written as

$$\frac{d}{dt}\sum_i Q_i^j = \sum_i \lambda_i^j - \sum_{i:Q_i^j(t)>0} \mu_i^j \cdot \sum_c G_{j,c}^{d(i)}(t), \forall c. \tag{5.1}$$

Theorem 5.1. *If $\mu_i^j = \mu_j \ \forall i$, then for any two different control strategies the sum of the queues will remain the same as long as $Q_i^j(t) > 0, \ \forall i$.*

Proof. Considering two different control strategies $u_j^A(t)$ and $u_j^B(t)$, and assuming that both will achieve the maximum departure rates μ_j, then using (5.1) the dynamics of the queues can be written as

$$\frac{d}{dt}\sum_i Q_i^j = \sum_i \lambda_i^j - \sum_i \mu_i^j \cdot \sum_c G_{j,c}^{d(i)}{}_{u_{A/B}}(t) = \sum_i \lambda_i^j - 2\mu_j. \tag{5.2}$$

It can be easily observed that the sum of the queues will depend only on their inflow rates λ_i such that if $\{Q_i^j > 0 \ \forall i, \ t\}$ then

$$\sum_i Q_i^j(t)_A = \sum_i Q_i^j(t)_B. \tag{5.3}$$

□

Note that we never assume $\sum_i \lambda_i^j > 2\mu_j$. Theorem 5.1 is valid as long as the queues are not empty.

Next, as a consequence of Theorem 5.1, we emphasise the following properties which will help us in the further developments.

Property 5.2. In saturated traffic, i.e. a situation where queues are never empty at green for all possible switching options to be envisaged, Theorem 5.1 says that it will not be possible to distinguish the quality of switching strategies on the basis of sum of queues. Hence control strategies should be compared on the basis of more complex criteria, e.g. sum of the squared queues $\frac{d}{dt}\sum_i {Q_i^j}^2 = 2(\sum_i \lambda_i^j - 2\mu_j) \cdot \sum_i Q_i^j$ or worst waiting times.

Property 5.3. The dynamics of the queues under the given conditions ($Q_i^j(t) > 0, \ \forall i, \ t$) will depend on the arrival rates $\lambda_i^j, i = 1, \ldots, 4$. In the case of an urban network, the arrival rates are approximately equal to the departure rates of the upstream intersections

of j. Consequently, if I_j and ${}^*I_j$ (the upstream intersection of I_j) could coordinate their flows, the performance may be better than the one of a fully distributed approach, where I_j and ${}^*I_j$ make local decisions regardless of their neighbours.

A major insight for our later development is represented by the next property. Although this property holds for saturated traffic, depending on the control choice made, we could leave the saturated traffic case and reach a lower departure rate.

Property 5.4. If at time t most of the queues $Q_j^i(t) = 0$, then the control strategy $u_j(t)$ that gives green to the directions where the most queues are zero will do worse than any other strategies.

Corollary 5.5. *Consider a fluid flow model with constant inflows for one intersection j, with constant λ_i^j and μ_j, with the approximation of stop-and-go traffic (no orange). Then, for any given switching strategy u_j, it is certain that applying the same switching x times faster, with any $x > 1$, will lead to a sum of queues situation that is at least as good as or better than with u_j. Moreover, if there exist two non-parallel directions i, k such that $\lambda_i^j + \lambda_k^j < \mu_j$, an optimal strategy to minimise the sum of the queues will have to ultimately switch faster and faster.*

Proof. Under the assumption of continuous average traffic conditions ($\lambda_j^i,\ \forall i$) and saturated traffic, the queues of a given direction will always be strictly positive at the instant when we switch it to green, since during the preceding red period - how short it may be - a queue has continuously built up. In other words, during $G_{j,0}^d$ in the directions opposite to where green is given, the queues grow no matter when $G_{j,0}^d$ ends. In the direction that has green, the queue decreases and could potentially get empty leading to the conclusion that it is better to switch fast enough to prevent the moment when a queue reaches zero.

The condition $\lambda_i^j + \lambda_k^j < \mu_j$ implies that the sum of queues in directions i and k decreases with time, namely by a fixed amount $|\lambda_i^j + \lambda_k^j - \mu_j|$ per time unit as long as none of these queues is empty; then, necessarily, for any optimal switching strategy (i.e. with maintaining all queues nonempty as long as possible), they will both approach zero and reach it in finite time. Now for a queue i of length ϵ, we can only give green during $\frac{\epsilon}{\mu_j - \lambda_i^j}$ to avoid that it reaches zero; the same applies to queue k. Since ϵ gets closer and closer to zero as the sum-of-queues constantly decreases, the green periods will get shorter and shorter. □

Property 5.6. A consequence of Corollary 5.5 is that the system would ultimately make an infinite number of switches in a finite amount of time (Zeno Phenomena). This behaviour would of course not have any practical advantage and is just a consequence of modelling over-abstraction (fluid with stop-and-go dynamics): in practice, yellow period makes switching faster and faster inefficient.

Property 5.7. As the traffic gets more and more loaded, that is λ_i^j increases w.r.t. μ_j, it takes longer and longer to empty an existing queue; thus there is less and less need to switch fast for avoiding queues to get empty. This is one explanation why in practice, cycle time gets larger as traffic intensity increases.

In practice, the lesson is that in open loop, switching faster is usually better, up to a modelling limit. Thus, we can use this fact as heuristic when predicting future switching behaviour.

Property 5.8. In order to deal with the modelling limit, we have to impose a minimal green time $\underline{G_j}$ between consecutive switching times as a hard constraint. The maximum green time $\overline{G_j}$ is not fundamentally necessary since probably the optimal controller will never reach this maximum bound and in principle a model without this constraint would work as well in practice. However, it is computationally practical to have this bound when exploring switching options. Furthermore, a maximum green period results in a maximum red period in conflicting direction, and a maximum red period is necessary for robustness against sensor failures.

The minimum green can also be obtained from optimisation if we include the yellow period in the platoon based model instead of using stop-and-go traffic.

5.2.2 A default switching rule for one intersection under very light traffic

Another extreme case study is for one intersection under very light traffic conditions ($\frac{\lambda_i^j}{\mu_i^j} \ll 1$). An optimal control strategy for this case is to give green whenever a vehicle arrives since the queues are (almost) always zero. The risk of having two simultaneous arrivals is negligible. The default state of the traffic light is irrelevant since it will react almost always to an arrival event.

5.2.3 A default switching rule for one intersection under intermediate traffic

In the case of a saturated intersection, the necessary control actions are to adjust cycle length, split and offset with the aim to switch as fast as possible. For the other extreme case, very light traffic conditions, these control actions are not really used. Next, for intermediate traffic conditions, there are significant random (but predictable according to the platoon based model) gaps between platoons. Flexibility in the control actions for this type of traffic is desirable and therefore a good approach would be to select actual switching times instead of using green cycle and split. In our approach, we use only switching times as control actions for intermediate traffic conditions.

Furthermore, coordination becomes more important between neighbouring intersections in the case of intermediate traffic conditions compared with the first two cases of traffic load. For the saturated traffic, the queues only depend on their inflows and not on the control strategies (see Theorem 5.1) whereas for very light traffic the coordination is not necessary since the queues are most of the time zero. A fully distributed solution for the intermediate case, where each local controller makes autonomous decisions based on the local observations is not desirable. A drawback of the fully distributed solution is that the local decisions do not take into account their effects on the neighbouring intersections. As opposed to a fully distributed solution, the computational burden of computing a centralised solution makes it infeasible even for a small network. Moreover, we expect that an urban network, no matter how small it is, will never be uniformly loaded (the assumption of the same traffic intensity at every intersection does not hold). Thus, dynamic green waves with respect to traffic changes are created by allowing the intersections less loaded to help the intersections heavily loaded. Applying the ”leader-follower” paradigm (a supervisor selects the heavily loaded intersections as leader agents and their neighbouring intersections as followers) and allowing communication between leaders and followers, the coordination can be achieved.

5.3 Distributed feedback control with leader-follower coordination framework for urban traffic networks

The urban traffic network $\aleph$ is controlled by locally selecting the switching times of its signalised intersections. The components of this network are the signalised intersections, the roads connecting intersections, and the sensors installed at several locations in the network along the roads of the network (including upstream and downstream of each intersection). Measurements of the successive passage time of vehicles are used to define platoons. Thus, each component of $\aleph$ can be modelled as a hybrid system, as described in Chapter 3, with platoons of vehicles moving along roads. Speed and size of platoons vary randomly as they move along roads. Queues form behind red traffic lights. The state of the network $X_{\aleph}(t)$, defined by (3.7), contains the location and size of all the platoons, and the size of the queues. In Chapter 4, we have developed estimators for urban traffic networks using the platoon based model introduced in Chapter 3. We proposed two particle filter algorithms for estimation of the queue sizes at intersections and the platoon locations along the links of the network.

Our coordinated control solution has a low layer formed by local control agents, and a supervisory layer $\mathbb{S}_{\aleph}$. We consider that every intersection I_j of $\aleph$ has an agent CA_j. Each agent uses a model based feedback controller to locally optimise its next switching times by minimizing a cost function J_j. The optimisation is performed every Δ_s seconds. The local optimisation problem will be described in detail in Section 5.3.1. Good performance requires that as little as possible of the capacity of the intersection I_j during green periods is wasted because there are no vehicles approaching the intersection. Green waves cannot be created and maintained if all agents independently select their optimal switching times. One possibility to achieve coordination starts by $\mathbb{S}_{\aleph}$ selecting the control agents of those intersections with the heaviest traffic load (where starvation causes the longest queues) as leaders. Thus, $\mathbb{S}_{\aleph}$ is able, using the "heaviest traffic load" criterion, to select among the control agents present in the network $\aleph$, the set $\mathbb{L}_{\aleph}$ of the leaders and the set $\mathbb{F}_{\aleph}$ of the followers. We denote the set of followers assigned to the leader agent $CA_j \in \mathbb{L}_{\aleph}$ as the set $\mathbb{F}_{\aleph}^{j} \subset \mathbb{F}_{\aleph}$. After all leaders run their local optimisation, then they communicate to their followers the optimal schedules found. The followers then run a local optimisation to compute their optimal schedule as a tradeoff between reaction to local traffic and

cost of not following the leader schedule. A follower is penalised with a cost C_{nf} if it uses a different schedule than what its leader sent. The not-following cost C_{nf} will be described in Section 5.3.2.

5.3.1 Control of a single intersection

Here we first consider as a subproblem the control of a single isolated intersection which gets information about arriving platoons. The agent CA_j controls the intersection I_j that has four entrance-exit pairs corresponding to the four directions indexed by i. We consider as known the following:

- all platoons approaching the intersection I_j at the current time t_0, between the upstream sensor and the intersection (we assume here the full state known but in practical control applications the controller will use the output of an estimator like the one developed in Section 4.3),
- the current queue lengths $Q_i^j(t_0), i = 1, 2, 3, 4$ at each of the 4 entrance points,
- the current green phase $G^d_{j,c=0}$ and the start time of the current green phase $\downarrow G^d_{j,c=0}$,
- the minimum length $\underline{G_j}$ and maximum length $\overline{G_j}$ allowed for the green phases.

Likewise, we denote the end time by $\uparrow G^d_{j,c}$ and the duration by $\updownarrow G^d_{j,c}$ of the c-th green phase.

For the given set-up, we want to find an optimal switching scenario $S_{j,\circledast}(t_0) = \{\uparrow G^d_{j,c}, \ldots\},\ c = 0, \ldots, \mathcal{N}_c - 1$ of the next $\mathcal{N}_c$ switching times that minimises a given cost criterion J_j over the time horizon T_h (where $\uparrow G^d_{j,N_c-1} \leq T_h$) at time t_0. For the rest of the presentation, we drop the time argument for $S_{j,\circledast}$. We denote with S_j the set of all possible switching scenarios at time t_0 over the time horizon T_h at intersection I_j. Since we want to minimise the waiting time over all queues, we propose the following

cost criterion

$$J_j\left(t_0, \uparrow G^d_{j,0}, \ldots, \uparrow G^d_{j,N_c-1}\right) = \sum_{i=1}^{4} \left(\underbrace{\int_{t_0}^{t_0+T_h} Q_i^j\left(t\right) dt}_{C_o-\text{optimisation cost}} + \underbrace{\alpha \cdot \frac{{Q_i^j}^2\left(T_h\right)}{2\mu_j}}_{C_f-\text{final cost}} \right) \tag{5.4}$$

where t_0 is the current time. More details about (5.4) follow in the next subsections.

In order to find the optimal schedule $S_{j,\circledast}$ of the next $\mathcal{N}_c$ switching times, we explore all the possible future switching scenarios S_j satisfying some constraints ($\underline{G_j} \leq\updownarrow G^d_{j,c} \leq \overline{G_j}$ for all $c = 0, \ldots, \mathcal{N}_c - 1$) over the given time horizon T_h. For each switching scenario $S_{j,z} \in S_j$, using the platoon based model, we compute its associated cost expressed by (5.4). We select the switching scenario $S_{j,\circledast}$ with the smallest cost. The resulting strategy $S_{j,\circledast}$ at time t is implemented just for one time step: either extend the current phase $G_{j,0}$, or switch. If the current green phase is at its maximum length $\overline{G_j}$, then we are forced to switch without optimisation anyway. At all times between a switch and minimum green $\underline{G_j}$, we do not switch nor optimise. Only if we are not in one of these cases, the optimisation is repeatedly carried out after Δ_s seconds. In fact, this is the description of model predictive control framework.

All the above remarks define the following optimisation problem

$$\text{minimize } J_j\left(t_0, \uparrow G_{j,0}, \underline{G_j}, \overline{G_j}, T_h, G_N\right) \tag{5.5}$$

subject to

$$\underline{G_j} \leq\updownarrow G^d_{j,c} \leq \overline{G_j}\ \forall d,\ c = 0, \ldots, \mathcal{N}_c.$$

Other constraints (e.g. $Q_i^j\left(t\right) < q_{max}, \forall t, i$) can be added to the optimisation.

Before going into detail on how to solve the proposed optimisation problem, we present the cost J_j. As can be observed from (5.4), the local cost criterion has 2 components for each entrance of the intersection j, called the C_o-*optimisation cost* and C_f-*final cost*, respectively.

5.3.1.1 The optimisation cost C_o

In order to evaluate C_o, the first part of (5.4), we look for a function of the evolution over time of the queue length at intersection I_j during the time interval t_0 to $t_0 + T_h$. At the beginning of this section we made the assumptions of knowing the full state $X_{L_l}(t),\ L_l \in {}^*I_j$ of the links carrying traffic towards I_j and using platoons to represent the traffic flow. Section 3 describes how the PBM models the platoons progression through an urban network. Since the PBM can be efficiently implemented as a discrete event system simulation tool, let us consider $\Delta T_m = \theta_m - \theta_{m^*}$ as the time difference between two consecutive events e_m, and e_{m^*}. The evolution of the automaton from Figure 3.10 represents the dynamics of a generic queue formed at any outflow boundary of a section or in the preselection lanes of an intersection. Each vehicle of a platoon progressively joints the queue $Q_i^j(t) : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ and modelled by the following positive piecewise-affine function

$$Q_i^j(t) = MAX\left(Q_i^j(t - \Delta T_m) + \frac{N}{(T_T - T_H)}\Delta T_m - \mu_j(exit_i) \cdot G_{j,c}^{d(i)}, 0\right) \tag{5.6}$$

that describes the state transitions of the automaton from Figure 3.10 (where T_H, T_T and N describe the first platoon merging with the queue $Q_i^j(t)$ and $exit_i$ models that the output may be blocked. A new event is generated in the platoon based model whenever a queue gets empty.)

We can derive now a formula for the evaluation of C_0

$$C_0 = \int_{t_0}^{t_o+T_h} Q_i^j(t)\,dt = \sum_{k=1}^{k^\diamond} \int_{t_0+\sum_{m=1}^{k-1}\Delta T_m}^{t_0+\sum_{m=1}^{k}\Delta T_m} Q_i^j(t)\,dt \tag{5.7}$$

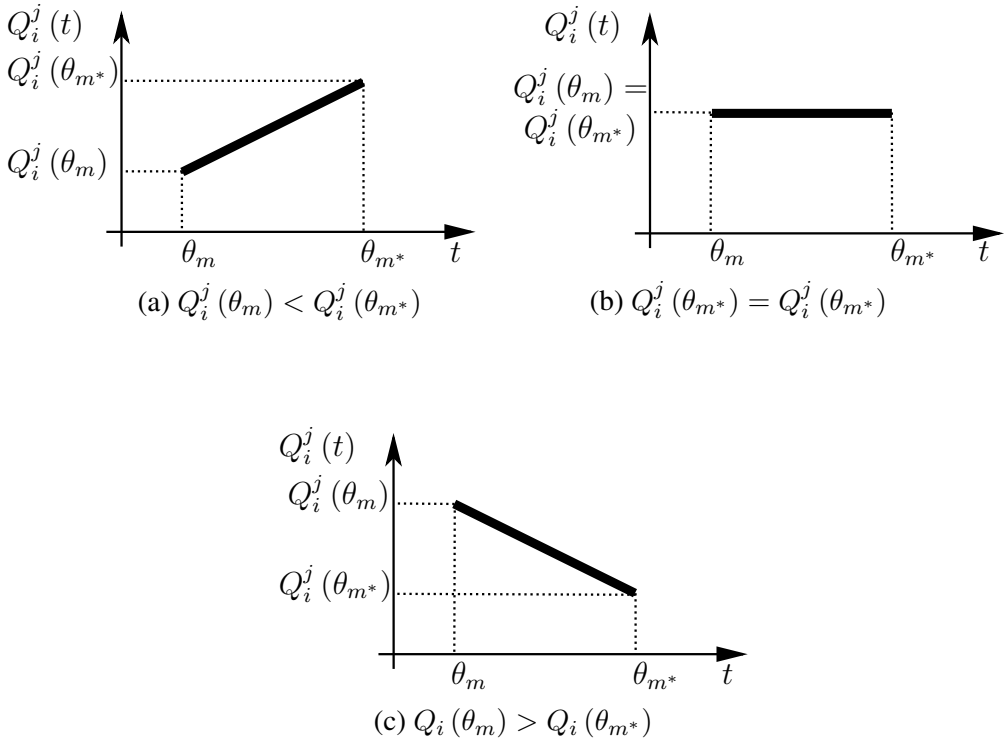
$$+ \begin{cases} \int_{t_0+\sum_{m=1}^{k}\Delta T_m}^{t_0+\sum_{m=1}^{k+1}\Delta T_m} Q_i^j(t)\,dt & \text{if } T_h = \sum_{m=1}^{k}\Delta T_m \\ \int_{t_0+\sum_{m=1}^{k}\Delta T_m}^{t_0+T_h} Q_i^j(t)\,dt & \text{otherwise,} \end{cases}$$

where $k^\diamond$ is the largest value of k such that $\sum_{m=1}^{k}\Delta T_m < T_h$.

The only part left to evaluate is $\int_{\theta_m}^{\theta_{m^*}} Q_i^j(t)\,dt,\ \forall\ Q_i^j(t) \geq 0$ as the area under the curves for the different cases shown in Figure 5.2 and has the following form

$$\int_{\theta_m}^{\theta_{m^*}} Q_i^j(t)\,dt = \frac{Q_i^j(\theta_m) + Q_i^j(\theta_{m^*})}{2} \cdot \Delta T_m. \tag{5.8}$$

The optimisation defined in (5.5) is rerun every Δ_s seconds by all the agents of the network, and before any possible switching time ($\Delta_s = 1$ second in our experiments presented at the end of the chapter). The selected sequence of switching times is obtained by simulating for a set of possible next switching times. In other words, the optimisation problem is solved by enumerating possible sequences of switching times, calculating for each one the performance, selecting the best one and implementing this best decision in case it requires immediate switching. The calculations could be further reduced by looking at fewer times (e.g. the times corresponding with the arrival or departure of the platoons, queue becoming zero, or when the length of the current maximum green phase $\overline{G_j}$ is reached). Further considerations about solving the optimization problem are presented in Section 5.3.1.4.



(a) $Q_i^j(\theta_m) < Q_i^j(\theta_{m^*})$ (b) $Q_i^j(\theta_{m^*}) = Q_i^j(\theta_{m^*})$

(c) $Q_i(\theta_m) > Q_i(\theta_{m^*})$

Figure 5.2: Cases of the local cost.

5.3.1.2 The final cost C_f

The final cost C_f accounts for the queue that remains at the end of the optimisation horizon. It models the fact that, when we stop our optimisation at T_h, having a larger end-queue means that the network will have to make more effort in the future (after the modelled horizon T_h), and this should be penalised. The importance of including the final cost in the optimisation cost is explained in [64].

The cost C_f, shown in Figure 5.3, is α times the time that it would take to deplete the residual queue after T_h if we would then directly switch to green for the direction $i+1$ and expressed by

$$C_f = \alpha \cdot \frac{{Q_{i+1}^j}^2(T_h)}{2\mu_j}. \tag{5.9}$$

The most natural value for α would be just 1, since we do not want to bias the strategies with a high C_0, but low C_f, or vice versa. Next, we show that even for a simple case, the

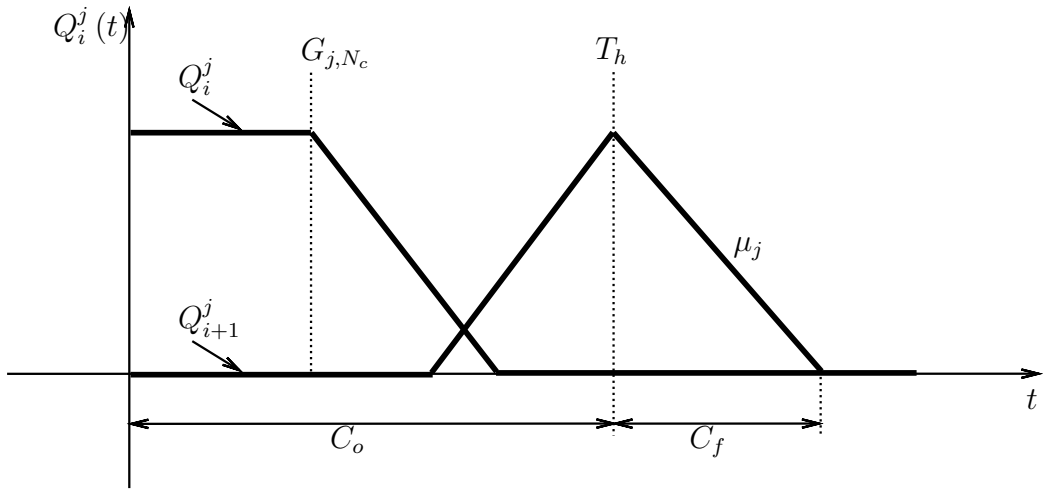


Figure 5.3: The penalisation of the queue that is not zero after the T_h.

cost criterion J_j is not necessarily a convex function of the switching times, depending on the parameters of platoon arrival times.

5.3.1.3 Optimisation cost case study

The easiest case to consider is an intersection with three inflows at constant average rate λ, and one precise inflow determined by the times T_a, T_b, and T_c at which platoons begin or end. Thus, we consider that between T_a and T_b a platoon arrives followed by a new platoon that begins to arrive at time T_c. In this example, the precise inflow is normalised such that when one platoon arrives, we consider an arrival rate of 1. If green, the vehicles instantaneously reach the maximum departure rate μ. As initial conditions, the queues for all 4 directions are zero and the directions with both inflows determined by the constat average rates have green. The switching time can be any moment between zero and T_f. Our goal is to show how much the optimal switching time, the shape of the cost function $J_j = C_o + C_f$, depends on every possible case of platoon arrivals (T_a, T_b, and T_c).

For the described set-up, we compute the cost function as the area under the curves plus the final costs (not represented on the figures), associated to three different switching times as follows:

- Case 1 - the switching time is T_a when the first platoon begins. The green phase

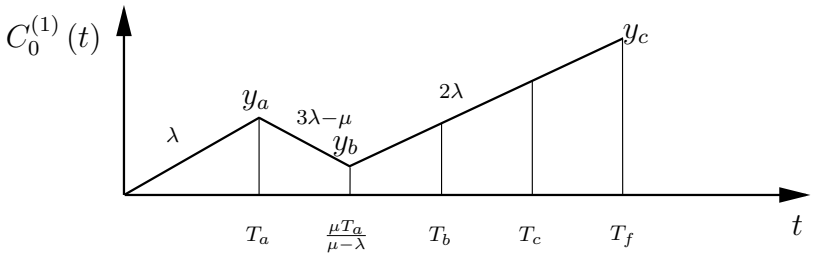


Figure 5.4: Switching time at T_a.

remains unchanged until T_f. As a consequence, platoons do not have to stop. The queue formed for the constant average rate inflow depletes until the time $\frac{\mu T_a}{\mu-\lambda}$. The cost $J_j^{(1)}$ is not influenced by the time T_b when the first platoon ends, nor time T_c when the second platoon begins. After the time $\frac{\mu T_a}{\mu-\lambda}$, the cost grows with a rate 2λ because the vehicles accumulate at a rate λ for each direction that has red.

The coordinate values that determine the cost are $y_a = \lambda T_a$, $y_b = \frac{2\lambda^2 T_A}{\mu-\lambda}$, and $y_c = 2\lambda(T_f - T_a)$. The cost for Case 1 is

$$J_j^{(1)} = \frac{\lambda(3T_a^2 - 4T_a T_f + 2T_f^2)}{2} + \frac{T_a^2\lambda^2}{2(\mu-\lambda)} + \alpha\frac{2\lambda^2(T_f - T_a)^2}{\mu}, \tag{5.10}$$

and represents the area under the curve shown in Figure 5.4 plus the final cost (not represented on the figure).

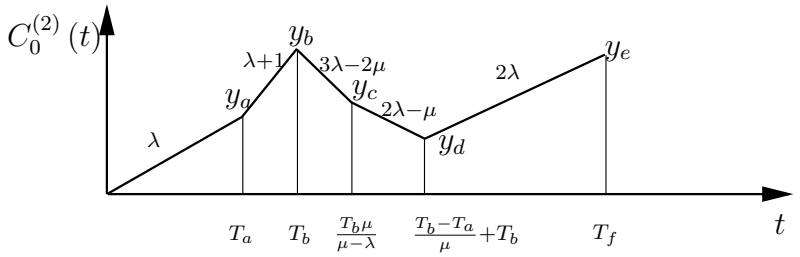


Figure 5.5: Switching time at T_b.

- Case 2 - another possible switching time that we consider is T_b, when the first platoon ends. Since between T_a and T_b the first platoon arrives at a rate of 1 and it is still red, then the cost increases at a rate $\lambda + 1$. If $T_b + \frac{T_b - T_a}{\mu} > T_b + \frac{T_b \lambda}{\mu - \lambda}$, the queue with the constant average inflow depletes first after a time $\frac{\lambda T_b}{\mu - \lambda}$ from T_b followed by the one with the precise inflow at a later time $T_b + \frac{T_b - T_a}{\mu}$. After the time $T_b + \frac{T_b - T_a}{\mu}$, the cost increases at the rate 2λ as in Case 1.

  The coordinate values that determine the cost are $y_a = \lambda T_a$, $y_b = T_b - T_a + \lambda Tb$, $y_c = T_b - T_a - \lambda T_b + \frac{\lambda^2 T_b}{\mu - \lambda}$, $y_d = \frac{2\lambda(T_b - T_a)}{\mu}$, and $y_e = 2\lambda(T_f - T_b)$. The cost $J_j^{(2)}$, as the area under the curve represented in Figure 5.5 plus the final cost (not represented on the figure), is

$$J_j^{(2)} = \frac{(3T_b^2\lambda)}{2} - T_a T_b + \lambda T_f^2 + \frac{T_a^2 + T_b^2}{2} - 2T_b T_f \lambda + \frac{(T_a - T_b)^2}{2\mu} + \frac{T_b^2 \lambda^2}{2(\mu - \lambda)} + \alpha \frac{2\lambda^2 (T_f - T_b)^2}{\mu}. \tag{5.11}$$

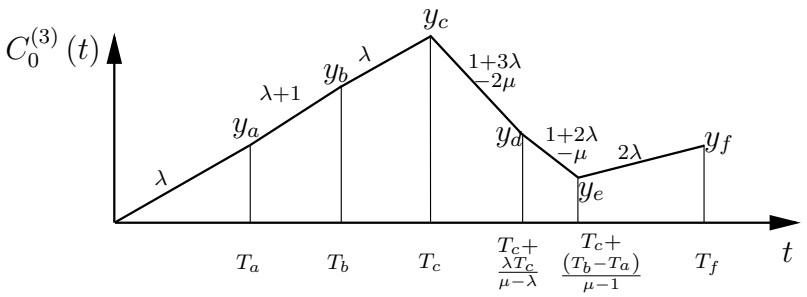


Figure 5.6: Switching time at T_c.

- Case 3 - another possible switching time that we investigate is T_c, when the second platoon begins. Until time T_b, $J_j^{(3)}$ is similar to the previous case, the cost increases at a rate $\lambda + 1$. Between T_b and T_c, the increase rate is only λ since the platoon ends at time T_b. After time T_c, the cost decreases at a rate $1 + 3\lambda - 2\mu$ because the second platoon arrives at a rate 1 and the first platoon still departs. If $T_c + \frac{\lambda T_c}{\mu - \lambda} < T_c + \frac{T_b - T_a}{\mu - 1}$, the queue with the constant average rate λ reaches zero at time $T_c + \frac{\lambda T_c}{\mu - \lambda}$ before the precise inflow queue that depletes

at a later time $T_c + \frac{T_b - T_a}{\mu - 1}$. After the time $T_c + \frac{T_b - T_a}{\mu - 1}$, both queues are depleted and the cost increases at a rate 2λ as in the other cases. The coordinate values that determine the cost are $y_a = \lambda T_a$, $y_b = T_b - T_a + \lambda Tb$, $y_c = T_b - T_a + \lambda T_c$, $y_d = T_b - T_a - T_c\lambda + \frac{T_c\lambda(\lambda+1)}{(\mu-\lambda)}$, $y_e = \frac{2\lambda(T_b - T_a)}{(\mu-1)}$, and $y_f = 2\lambda(T_f - T_c)$. The cost $J_j^{(3)}$, as the area under the curve represented in Figure 5.6 plus the final cost (not represented on the figure), is

$$\begin{aligned} J_j^{(3)} = & \frac{T_c^2\mu^2}{2(\mu-1)} - \frac{T_aT_b - T_aT_c + T_bT_c + T_b^2 + \lambda T_c^2 + \lambda T_f^2}{(\mu-1)} \\ & + \frac{\mu(T_a^2 - T_b^2 + T_c^2 + \mu T_c^2}{2(\mu-1)} + \frac{2T_cT_f\lambda - T_aT_c\mu + T_bT_c\mu}{(\mu-1)} \\ & + \frac{\lambda\mu T_c^2 + \lambda\mu T_f^2 - 2\lambda\mu T_cT_f}{(\mu-1)} + \alpha\frac{2\lambda^2(T_f - T_c)^2}{\mu}. \end{aligned} \tag{5.12}$$

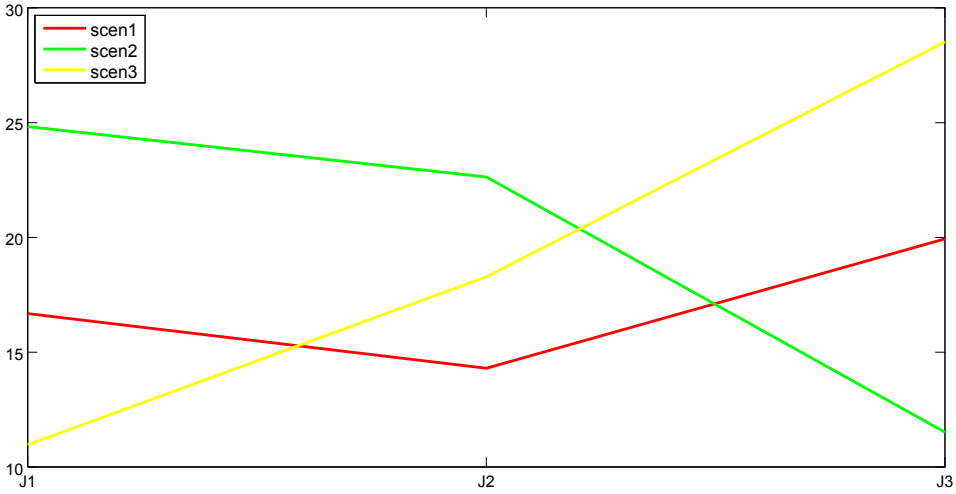


Figure 5.7: The cost can be convex or not depending on platoon arrival times.

Based on the described cases with $\lambda = 0.2$, $\mu = 1.5, \alpha = 1$ and $T_f = 10$, we investigate three different scenarios for the precise inflow direction as follows:

- scen1 - $T_a = 2,\ T_b = 4$, and $T_c = 7$,
- scen2 - $T_a = .1,\ T_b = .6$, and $T_c = 7$,

- scen3 - $T_a = 4,\ T_b = 7.5$, and $T_c = 8$,

and the results are centralised in Table 5.1.

	$J_j^{(1)}$	$J_j^{(2)}$	$J_j^{(3)}$
scen1	16.67	14.29	19.93
scen2	24.83	22.63	11.50
scen3	10.96	18.28	28.52

Table 5.1: The obtained cost values for the three different scenarios

Figure 5.7 shows the costs for three different values of $T_a,\ T_b$, and T_c and for each switching time. Even for this simple example with 3 constant average rate inflows, the cost is not convex in the variable "switching time". The cost for choice T_b is sometimes lower than for the choice T_a or T_c (convex like in scen1) and sometimes not, depending on the parameters of platoon arrival times. Computing the cost for a given scenario and input choice is not difficult, even analytically. But, because of the various non-convex cost possibilities, it is difficult to find which of all the possible input choice is optimal. We see on Figure 5.7 that the cost function can have basically all different shapes depending on platoon arrivals. It thus seems difficult to apply a more efficient strategy for finding the optimal switching time, than actually computing the value of the cost for all possibilities. We will therefore implement the optimisation by indeed enumerating all possible cases and selecting the best one.

5.3.1.4 Solving the optimisation problem

The optimisation problem introduced in Section 5.3.1 must be solved in real-time by the control agent CA_j placed in intersection I_j. As we already mentioned, we use the model predictive control framework by optimizing over a finite time-horizon and only implementing the current decision. The current decision extends or ends the green for the current direction $G^d_{j,0}$, while taking into account future switching times $\{\uparrow G^d_{j,c}, \ldots\},\ c = 1, \ldots, \mathcal{N}_c - 1$.

To explore each time all the possible switching scenarios can become infeasible very fast due to the high number of scenarios. The maximum number of scenarios for the

first switching point is given by $(\overline{G_j}-\underline{G_j})/\Delta_s$. For more switching points, over a longer time horizon T_h, the number of scenarios has an exponential growth ($\left(\frac{\overline{G_j}-\underline{G_j}}{\Delta_s}\right)^{\mathcal{N}_c}$ for $\mathcal{N}_c$ switching times). Usually the links for which we have data are short such that the information about the incoming platoons does not cover more than one switching time plus the mandatory $\underline{G_j}$ afterwards. In fact our time horizon is very limited and so not too many switching scenarios can occur for $\Delta_s = 1[sec.]$.

If T_h is large, there will be many switching options, whereas a small T_h leads to only one switching option after which we must wait for the minimum green phase $\underline{G_j}$. Another important factor for choosing T_h relates to how long in the future we know the incoming traffic from the links upstream intersection I_j. This last element has to do with the length δ_l of the links. It does not make sense to optimise for a very long T_h (e.g. hours) if the information about the platoons entering on the links is for the next dozens of seconds. In the opposite case ($T_h \ll \delta_l$), the anticipation effect is lost. Thus, the optimisation time horizon T_h has to be greater than the time delay δ_l of the longest link of I_j but not too much since after δ_l we have to replace the information about platoons with average traffic and hence we loose the real-time advantage. The importance of choosing a longer time horizon T_h than the time delay δ_l is also explained in [64] as a reaction to a rolling horizon switching strategy for a single intersection proposed in [60].

It is a drawback of the pure leader-follower framework, that a leader can only react to traffic, but not to future plans of the followers: in fact, the leader itself imposes these future plans. If feedback from the followers' plans towards the leader decisions were allowed, we could exploit a further coordination mechanism where each leader takes into account the times at which upstream followers intend to send traffic towards it in the future. In the current proposal, each leader assumes that the followers will perfectly execute whatever plan is assigned to them. Instead, one could consider that followers also communicate their preferences towards the leaders. This would create a full feedback loop in the decision taking strategy, so the latter would have to be reformulated to avoid a chicken-and-egg cycle. In general, strategies where the intersections would influence each other more bidirectionally are worth investigating in future work.

The computation of δ_l for a link L_l depends on the length of the link and the used

speed. The length of the links in urban traffic networks is in general short especially for European road networks. In the description of the platoon based model, the speed v_l is independently randomly chosen from a given probability distribution. Selecting the highest legal speed to compute δ_l, a planned switch could be too early for unexpectedly slow vehicles, so the latter arrive just after the switching and would have to wait for a full red half-cycle. Conversely, taking a too low speed means that the first vehicle will have to wait (to slow down in practice) for one second. This is not too bad for the cost that we compute, (and also the real cost) compared to the cost of waiting for a full red half-cycle. In practice, if the highest speed is used to compute δ_l, the driver may see orange and speed up (not in the case of our model, leading to a higher cost) which ultimately could not be safe. Thus, we consider a speed close to the minimum speed value for the computation of δ_l.

Algorithm 5.1 CA_j computes at time t_0 the time $G^d_{j,0}$ of the next switch of the traffic light of intersection I_j

Function DistController$\Big(\text{in} : t_0, \downarrow G^d_{j,0}, \underline{G_j}, \overline{G_j}, T_h, G_N\Big)$ (out : $S_j, J_j(S_j)$)
$n_{scen} := \frac{\downarrow G^d_{j,0} + \overline{G_j} - t_0}{\Delta_s}$
for $z = 1$ to n_{scen} **do**
  $G_{j,0} := t_0 + z\Delta_s$
  $S_{j,z} := \{\uparrow G^d_{j,0}, \uparrow G^d_{j,0} + G_N, \uparrow G^d_{j,0} + 2G_N, ..., \uparrow G^d_{j,0} + mG_N\}$
  compute $J_j\left(S_{j,z}, T_h\right)$ using PBM
end for
return $S_j, J_j(S_j)$
EndFunction

In conclusion, we only optimise the first switching time $\uparrow G^d_{j,0}$ and consider nominal green periods G_N for the next ones. A general switching scenario is $S_{j,z} = \{\uparrow G^d_{j,0}, \uparrow G^d_{j,0} + G_N, \uparrow G^d_{j,0} + 2G_N, ..., \uparrow G^d_{j,0} + mG_N\}$ with $\uparrow G^d_{j,0} + mG_N \leq t_0 + T_h$. The reason why we look one switching time ahead is due to the fact that the considered links are so short such that after the first switching time there is no more information about the arriving platoons. The value G_N is a nominal value selected offline as a function of the expected volume of traffic and the green length at the steady state from the upstream intersection. In other words, at the current time t_0, agent CA_j decides to extend or not the current green phase based on:

- the local information from the intersection I_j,
- the platoons that already entered on the links that lead to the intersection I_j up to time $t_0 + \delta_l$,
- the average traffic used to replace the missing information after time $t_0 + \delta_l$ up to $t_0 + T_h$,
- considering the next switching times using a nominal period of time G_N.

We first describe a fully distributed algorithm that is only a building block of our leader-follower development.

Algorithm 5.2 The control agent CA_j of intersection j determines if it is suitable to switch now or to extend the current green

1 **Function CA** $(\text{in} : j, t_0)$
2 $\downarrow G^d_{j,0}, \underline{G_j}, \overline{G_j}, T_h, G_N$
3 **if** $t_0 \geq \downarrow G^d_{j,0} + \underline{G_j}$ **then**
4 $(S_j, J_j(S_j)) :=$ **DisController** $\left(t_0, \downarrow G^d_{j,0}, \underline{G_j}, \overline{G_j}, T_h, G_N\right)$
5 select $S_{j,\circledast}$ with the minimum $J_j(S_j)$
6 **if** $\uparrow G^d_{j,0} = t_0$ **then**
7 switch the traffic light of I_j
8 **end if**
9 **end if**
10 **EndFunction**

All the above remarks and conclusions lead to a simpler and therefore faster optimisation problem. Algorithm 5.1 returns the set of all feasible switching scenarios S_j and their associated cost $J_j(S_j)$ for agent CA_j at time t_0. The cost of each scenario, $S_{j,z}$ is computed by running the platoon based model with the given switching scenario from time t_0 until T_h (line 6 in Algorithm 5.1). The control agent CA_j deployed in intersection I_j calls Algorithm 5.2 to determine if the current green phase has to be terminated or not. In this way, each agent of the urban traffic network $\aleph$ is able to locally optimise its next switching time. What is left to define is the not-following cost showing the effects of the switching times between neighbouring intersections.

5.3.2 The not-following cost C_{nf}

The control agent $CA_j,\ CA_j \in \mathbb{L}_{\aleph}$, selected as leader, sends to its followers the optimal schedule found by CA_j *shifted backward in time with* δ_l. The desired schedule sent by the leader to its follower(s) has a physical interpretation, namely the time interval(s) when the platoon(s) should arrive at the leader. The follower has to pay a cost, C_{nf}, added to the local optimisation cost if it wants to make a different schedule than the desired schedule send by its leader.

We choose the cost C_{nf} as a function of the queue formed at the leader under average traffic due to the deviation of the follower from the desired schedule. Thus, for computing C_{nf}, we use average traffic (average arrival λ, and average departure μ) since anything else seems too complex for real-time computation. For simplicity, we drop the indices i, j from λ and μ (they can later be inserted for each leader-follower pair). We will compute how leader queues would evolve over one nominal green period G_N for average traffic and with a starting queue zero at leader and follower. These can be justified, because unless average traffic is so high that it leads to saturation and thus network instability ($2\lambda > \mu$)

- a proper nominal schedule will ultimately drive any initial queues to zero after a few cycles,
- and once the queues are zero at the end of a green cycle, under average traffic it is certain that they will also be zero at the end of the next green cycle (this is readily observed by computing C_{nf} under these assumptions, as we do in the following pages).

Furthermore as already mentioned, the links for which we have data are short such that the information about the incoming platoons does not cover more than one switching time plus the mandatory $\underline{G_j}$ afterwards. We could optimise over more switching times by assuming some traffic model, but since we have no data, we expect that doing so would not give a significant improvement as opposed to just taking the nominal green period. Thus, it makes sense to use the nominal green periods G_N for the average traffic used to replace the missing information after δ_l (the delay between a leader and its follower) up to T_h. The parameters λ and μ are selected based on historical measurements. The cost C_{nf} is valid only if $2\lambda < \mu$.

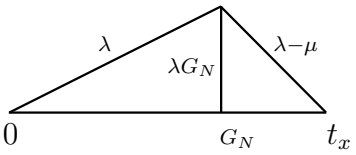


Figure 5.8: Computation of the t_x time when the queue goes to 0

When the follower switches to green for the leader's direction, while the queue gets empty, the vehicles arrive to the leader at a rate μ (the departure rate at the follower). The vehicles continue to arrive at a rate λ (equal with the rate at which the vehicles arrive at the follower) after the queue is empty. We denote the time when the queue at the follower is empty as $t_x = \frac{\lambda G_N}{\mu-\lambda}$. Figure 5.8 presents this situation.

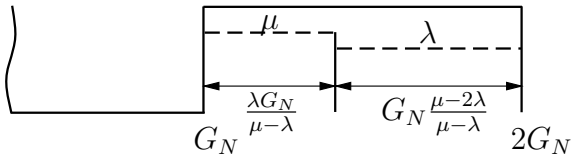


Figure 5.9: Average traffic during a nominal green period G_N

Using t_x, we can characterise the average traffic at the leader coming from the follower during a nominal green period G_N as shown in Figure 5.9. The traffic arrives at rate μ until t_x and at a rate λ for the rest of the green period $G_N \frac{\mu-2\lambda}{\mu-\lambda}$.

We define the deviation η between the desired schedule (send by the leader and shifted backward in time with δ_l) and the schedule of the follower as the time difference between the moments when the leader and its follower switch to green. In order to create and maintain the green waves it is important that the follower(s) of a leader reduce its(their) phase shift(s) (i.e $\eta \to 0$). We will show now, for different values of η, the effect (in terms of queue lengths) of the follower at the leader. In each case, we compute the area under the graph defined by the queue formed at the leader as follows:

1. $\eta = 0$ - the leader and its follower are fully aligned and the traffic sent by the follower will catch the green light at the leader without destroying the platoon created at the follower during the red period. As can be observed in Figure 5.10, the queue at the leader is 0 if the leader and the follower are in phase. Thus, the area under the graph is $\mathbb{A}_1 = 0$.

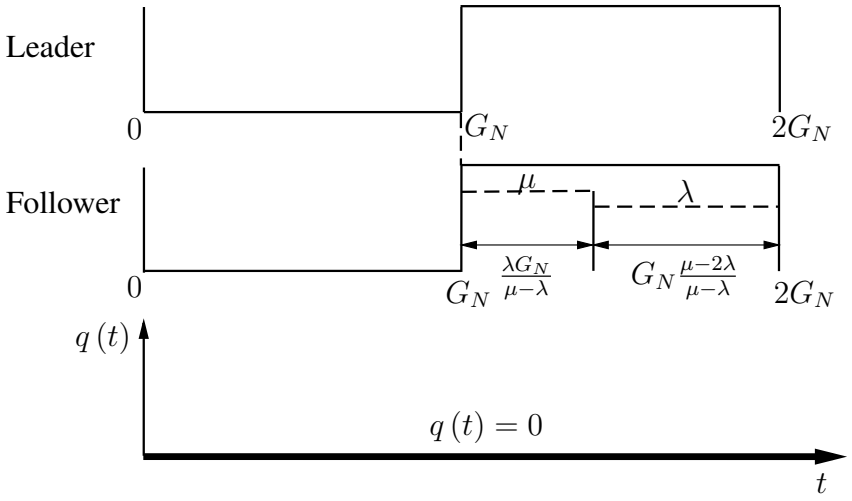


Figure 5.10: For $\eta = 0$ the follower is in phase with its leader and therefore there is not accumulated queue at the leader.

2. if $0 < \eta < \frac{\lambda G_N}{\mu-\lambda}$ (for $\lambda \leq \left(1-\frac{\sqrt{2}}{2}\right)\mu$) or $0 < \eta < G_N\left(1-\frac{2\lambda}{\mu}\right)$ (for $\lambda > \left(1-\frac{\sqrt{2}}{2}\right)\mu$) , the accumulated queue at the leader looks like the one in Figure 5.11.

   Thus, the area under the graph can be computed as

$$\mathbb{A}_2 = 0.5\eta^2\mu + \mu\eta\left(\frac{\lambda G_N}{\mu-\lambda} - \eta\right) + 0.5\frac{\eta^2\mu^2}{\mu-\lambda}, \tag{5.13}$$

   where $T_2 = G_N\frac{\mu}{\mu-\lambda} + \eta\frac{\lambda}{\mu-\lambda}$ with $(T_2 < 2G_N - \eta)$.

3. if $\frac{\lambda G_N}{\mu-\lambda} < \eta < G_N\left(1-\frac{2\lambda}{\mu}\right)$ (for $\lambda \leq \left(1-\frac{\sqrt{2}}{2}\right)\mu$) or $G_N\left(1-\frac{2\lambda}{\mu}\right) < \eta < \frac{\lambda G_N}{\mu-\lambda}$ for $\lambda > \left(1-\frac{\sqrt{2}}{2}\right)\mu$, we can distinguish two possible cases:

   3.a The queue at the leader is zero before the time $T_{3a} < 2G_N - \eta$, as in Figure 5.12, where $T_{3a} = G_N\left(\frac{2\lambda-\mu}{\lambda-\mu}\right) + \eta\left(\frac{\lambda}{\lambda-\mu}\right)$. This case occurs if $\lambda \leq \left(1-\frac{\sqrt{2}}{2}\right)\mu$ for $\frac{\lambda G_N}{\mu-\lambda} < \eta < G_N\left(1-\frac{2\lambda}{\mu}\right)$.

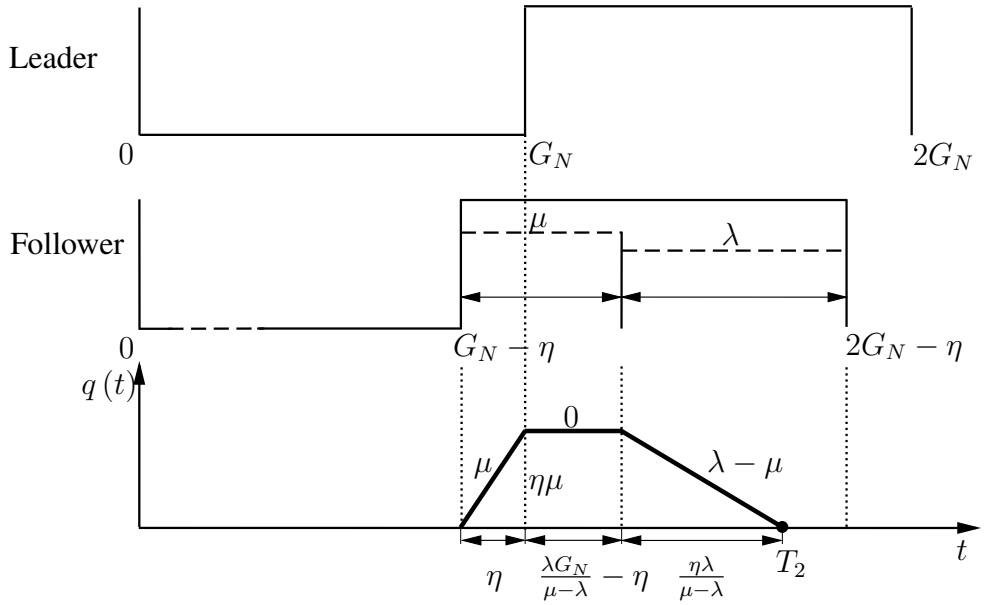


Figure 5.11: Case 2: if $0 < \eta < \frac{\lambda G_N}{\mu-\lambda}$ (for $\lambda \leq \left(1 - \frac{\sqrt{2}}{2}\right)\mu$) or $0 < \eta < G_N\left(1 - \frac{2\lambda}{\mu}\right)$ (for $\lambda > \left(1 - \frac{\sqrt{2}}{2}\right)\mu$)

The area under the graph has the following form

$$\mathbb{A}_{3a} = 0.5\left(\frac{G_N\lambda\mu}{\mu-\lambda} + G_N\lambda + \eta\lambda\right)\left(\eta - \frac{G_N\lambda}{\mu-\lambda}\right) + 0.5\mu\left(\frac{G_N\lambda}{\mu-\lambda}\right)^2 + 0.5\frac{(G_N+\eta)^2\lambda^2}{\mu-\lambda}. \quad (5.14)$$

3.b The queue at the leader is zero after the time $2G_N - \eta < T_{3b} < 2G_N$ as in Figure 5.13, where $T_{3b} = G_N\left(1 + \frac{2\lambda}{\mu}\right)$. This case occurs if $\lambda > \left(1 - \frac{\sqrt{2}}{2}\right)\mu$ for $G_N\left(1 - \frac{2\lambda}{\mu}\right) < \eta < \frac{\lambda G_N}{\mu-\lambda}$.
The area under the graph is

$$\mathbb{A}_{3b} = 0.5\mu\eta^2 + 0.5\frac{G_N(\mu-2\lambda)}{\mu-\lambda}(2\mu\eta - G_N\mu + 2G_N\lambda) + \mu\eta\left(\frac{G_N\lambda}{\mu-\lambda} - \eta\right) + 0.5\frac{(\mu\eta - G_N\mu + 2G_N\lambda)^2}{\mu}. \quad (5.15)$$

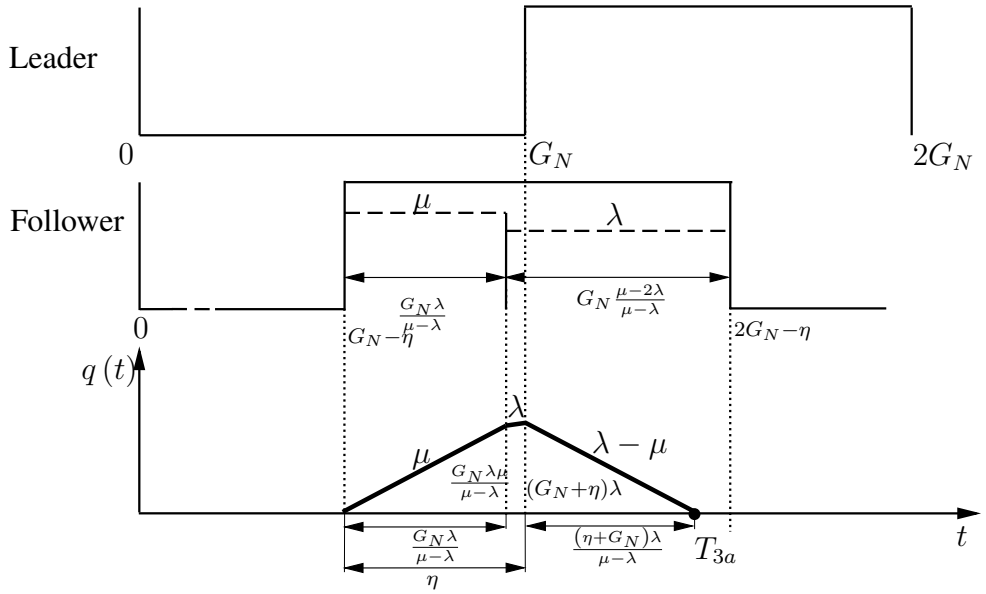


Figure 5.12: Case 3a: if $\frac{\lambda G_N}{\mu-\lambda} < \eta < G_N\left(1-\frac{2\lambda}{\mu}\right)$ (for $\lambda \leq \left(1-\frac{\sqrt{2}}{2}\right)\mu$)

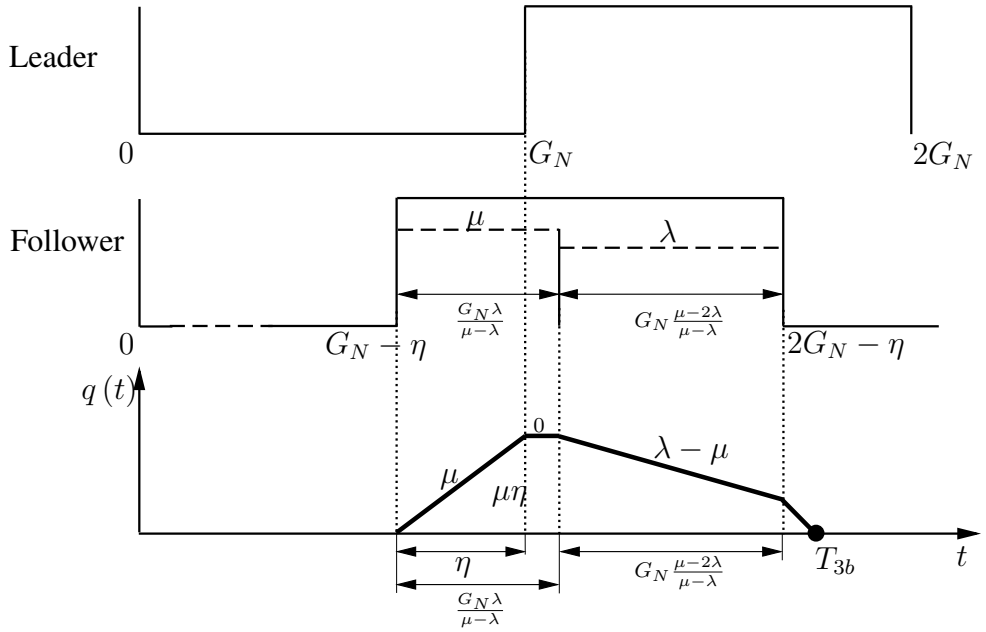


Figure 5.13: Case 3b: if $G_N\left(1-\frac{2\lambda}{\mu}\right) < \eta < \frac{\lambda G_N}{\mu-\lambda}$ (for $\lambda > \left(1-\frac{\sqrt{2}}{2}\right)\mu$)

4. if $G_N\left(1-\frac{2\lambda}{\mu}\right) < \eta < G_N$ (for $\lambda \leq \left(1-\frac{\sqrt{2}}{2}\right)\mu$) or $\frac{\lambda G_N}{\mu-\lambda} < \eta < G_N$ (for $\lambda > \left(1-\frac{\sqrt{2}}{2}\right)\mu$), the accumulated queue at the leader looks like the one in Figure 5.14. The queue is zero at time $T_4 = G_N\left(1+\frac{2\lambda}{\mu}\right)$ similar with T_{3b}.

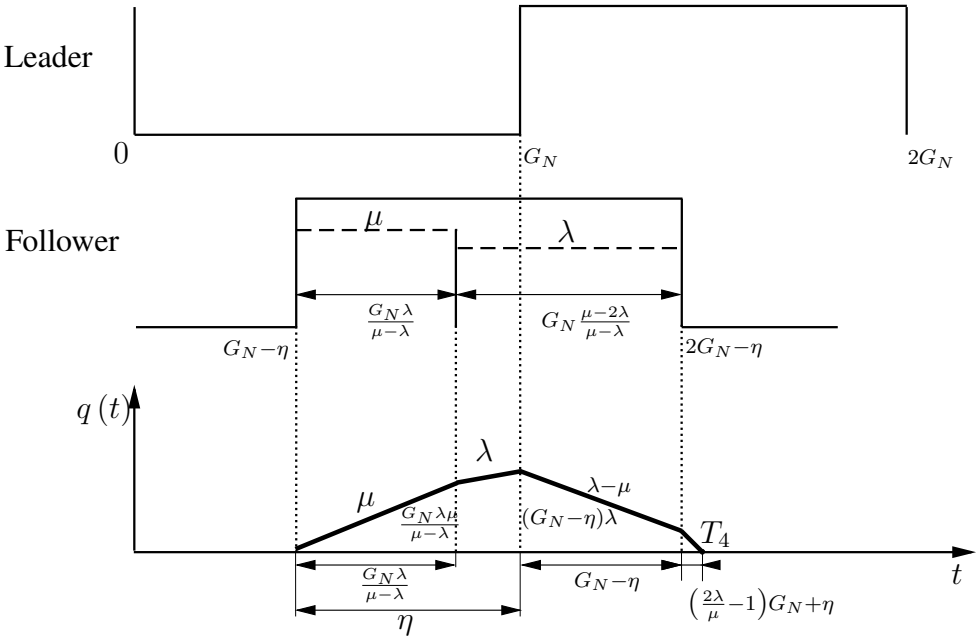


Figure 5.14: Case 4: if $G_N\left(1-\frac{2\lambda}{\mu}\right) < \eta < G_N$ (for $\lambda \leq \left(1-\frac{\sqrt{2}}{2}\right)\mu$) or $\frac{\lambda G_N}{\mu-\lambda} < \eta < G_N$ (for $\lambda > \left(1-\frac{\sqrt{2}}{2}\right)\mu$)

Thus, the area under the graph is

$$\begin{aligned}\mathbb{A}_4 = {} & 0.5\mu\left(\frac{G_N\lambda}{\mu-\lambda}\right)^2 + 0.5\left(G_N-\eta\right)\left(\mu\eta-\mu G_N+3\lambda G_N+\eta\lambda\right) \\ & 0.5\left(\eta-\frac{G_N\lambda}{\mu-\lambda}\right)\left(\frac{G_N\lambda\mu}{\mu-\lambda}+G_N\lambda+\eta\lambda\right)+ \\ & 0.5\frac{\left(\mu\eta-G_N\mu+2G_N\lambda\right)^2}{\mu}. \end{aligned} \tag{5.16}$$

5. if $G_N < \eta < G_N + \frac{G_N\lambda}{\mu-\lambda}$ the accumulated queue at the leader looks like the one in Figure 5.15. The queue is zero at time $T_5 = G_N\left(2+\frac{2\lambda}{\mu}\right) - \eta$ (with $T_5 < 3G_N - \eta < 2G_N$) .

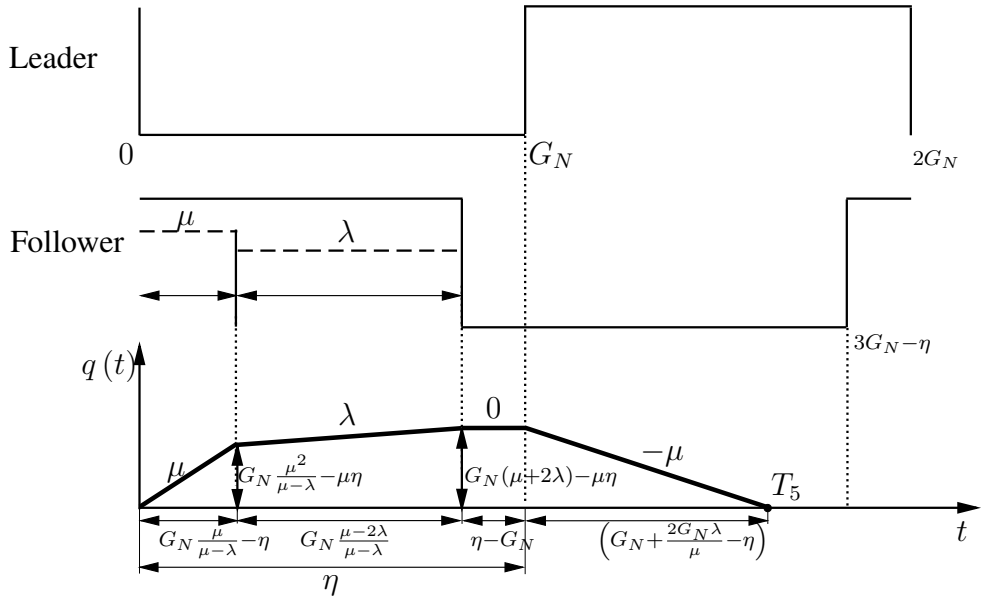


Figure 5.15: Case 5: if $G_N < \eta < G_N + \frac{G_N\lambda}{\mu-\lambda}$

Thus, the area under the graph is

$$\mathbb{A}_5 = 0.5\mu\left(\frac{G_N\mu}{\mu-\lambda}-\eta\right)^2 + 0.5G_N\frac{\mu-2\lambda}{\mu-\lambda}\left(\frac{G_N\mu^2}{\mu-\lambda}-2\eta\mu+G_N\mu+2G_N\lambda\right)$$
$$+\,(\eta-G_N)\,(G_N\mu+2G_N\lambda-\mu\eta)+0.5\mu\left(G_N+\frac{2G_N\lambda}{\mu}-\eta\right)^2. \quad (5.17)$$

6. if $\frac{\mu G_N}{\mu-\lambda} < \eta < 2G_N$, the accumulated queue at the leader is shown in Figure 5.16. The queue is zero at time $T_6 = G_N\left(1+\frac{2\lambda}{\mu}\right)-\eta\frac{\lambda}{\mu}$ (with $T_6 < 3G_N-\eta < 2G_N$).

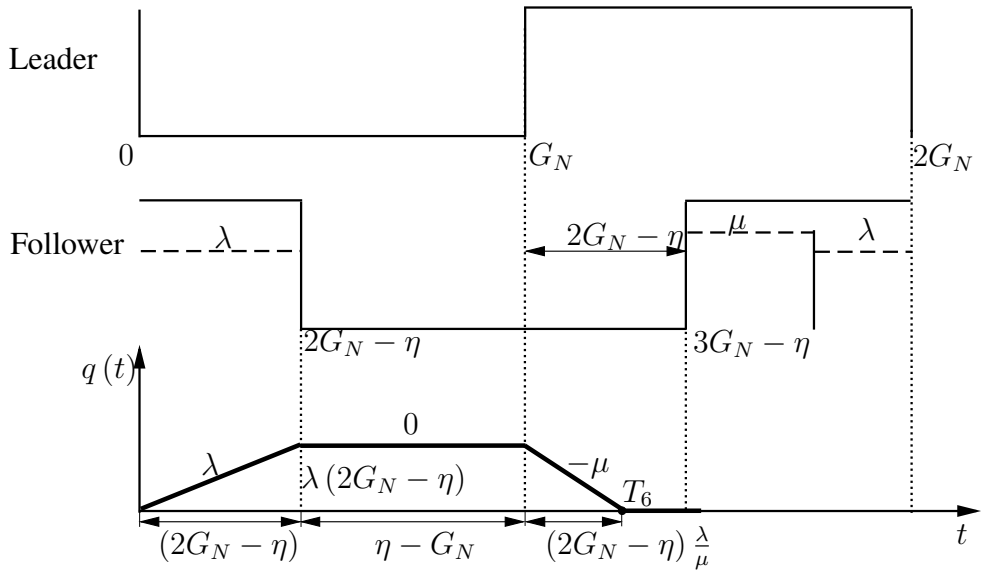


Figure 5.16: Case 6: if $\frac{G_N\mu}{\mu-\lambda} < \eta < 2G_N$

In this last case, the area under the graph is

$$\mathbb{A}_6 = 0.5\lambda \left(2G_N - \eta\right)^2 + \frac{\lambda\mu \left(2G_N - \eta\right)\left(\eta - G_N\right) + 0.5 \left(2G_N - \eta\right)^2 \lambda^2}{\mu}. \tag{5.18}$$

Using the above defined formulas, we are able to compute the cost C_{nf} for not following the leader's desired schedule by a follower for $\eta \in [0, 2G_N]$. Algorithm 5.3 describes how this cost is computed.

Algorithm 5.3 The $CompCnf\ (\eta, G_N, \lambda, \mu)$ function computes the penalisation of the follower for not following the leader

Function CompCnf $(\text{in} : \eta, G_N, \lambda, \mu)\ (\text{out} : C_{nf})$
$mincond := MIN\left(\frac{G_N\lambda}{\mu-\lambda}, G_N\left(1 - \frac{2\lambda}{\mu}\right)\right)$
$maxcond := MAX\left(\frac{G_N\lambda}{\mu-\lambda}, G_N\left(1 - \frac{2\lambda}{\mu}\right)\right)$
if $0 < \eta < mincond$ **then**
  $C_{nf} := \mathbb{A}_2$ as in (5.13)
else if $mincond \leq \eta < maxcond$ **then**
  if $\lambda \leq \left(1 - \frac{\sqrt{2}}{2}\right)\mu$ **then**
    $C_{nf} := \mathbb{A}_{3a}$ as in (5.14)
  else
    $C_{nf} := \mathbb{A}_{3b}$ as in (5.15)
  end if
else if $maxcond \leq \eta < G_N$ **then**
  $C_{nf} := \mathbb{A}_4$ as in (5.16)
else if $G_N < \eta \leq \frac{G_N\mu}{\mu-\lambda}$ **then**
  $C_{nf} := \mathbb{A}_5$ as in (5.17)
else if $\frac{G_N\mu}{\mu-\lambda} \leq \eta < 2G_N$ **then**
  $C_{nf} := \mathbb{A}_6$ as in (5.18)
else
  $C_{nf} := 0$
end if
Return C_{nf}
EndFunction

An example of the output of Algorithm 5.3 with $\mu = 0.5,\ G_N = 32$, and two different lambda values $\lambda_1 = 0.2,\ \lambda_2 = 0.1$ is shown in Figure 5.17.

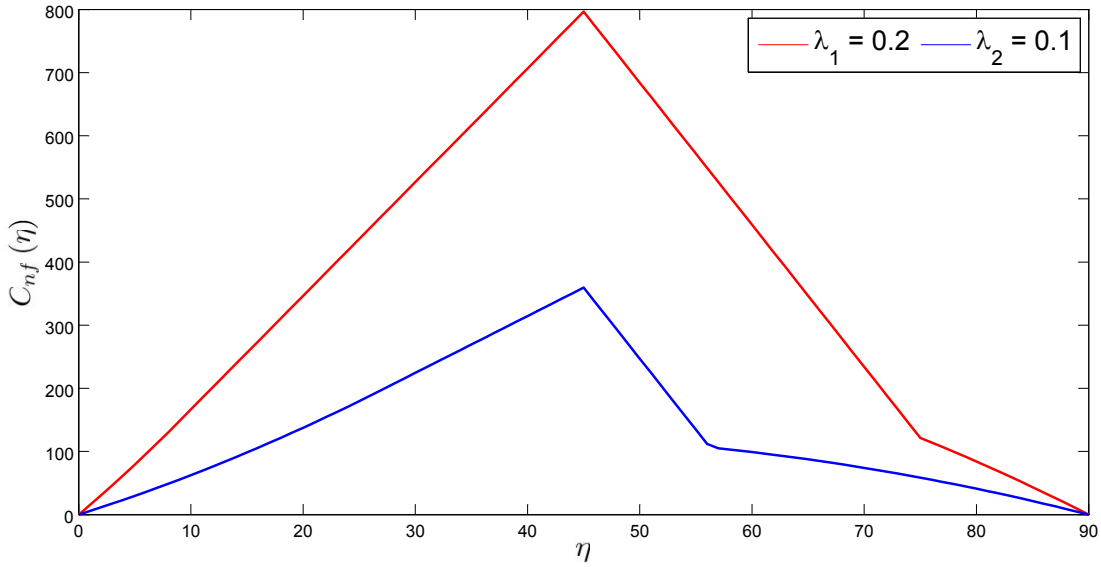


Figure 5.17: The $C_{nf}(\eta)$ for $\mu = 0.5,\ G_N = 32$, and $\lambda_1 = 0.2,\ \lambda_2 = 0.1$.

The two values λ_1 and λ_2 are chosen to show the two possible cases $\mathbb{A}_1 \rightarrow \mathbb{A}_2 \rightarrow \underline{\mathbb{A}_{3b}} \rightarrow \mathbb{A}_4 \rightarrow \mathbb{A}_5 \rightarrow \mathbb{A}_6$, $\mathbb{A}_1 \rightarrow \mathbb{A}_2 \rightarrow \underline{\mathbb{A}_{3a}} \rightarrow \mathbb{A}_4 \rightarrow \mathbb{A}_5 \rightarrow \mathbb{A}_6$, respectively.

5.3.2.1 Leader-Follower control algorithm

The main difference between the fully distributed controller presented in Algorithm 5.2 and the leader-follower framework described in Algorithm 5.4 stands in the coordination between the leaders and the followers. In the case of a leader, the optimal switching scenario $S_{j,\circledast}$ (denoted $S^L_{j',\circledast}$ from the follower's perspective) found is sent to all its followers. Once a follower receives the desired schedule from its leader, it computes the deviation η for all its local feasible scenarios at time t_0. The not-following cost that corresponds to a feasible scenario is added to the cost of that scenario. In this way, the leaders coordinate their followers and try to create green waves between neighbouring intersections.

Algorithm 5.4 CA_j of intersection I_j in Leader-Follower.

Function CA (in : j, t_0)
$\downarrow G^d_{j,0}, \underline{G_j}, \overline{G_j}, T_h, G_N, G_N, \lambda, \mu, S^L_{j',\circledast} = S_{j,\circledast}$
if $t_0 \geq \downarrow G^d_{j,0} + \underline{G_j}$ **then**
    $(S_j, J_j) :=$ **DisController** $\left(t_0, \downarrow G^d_{j,0}, \underline{G_j}, \overline{G_j}, T_h, G_N\right)$
end if
if $CA_j \in \mathbb{L}_{\aleph}$ and $CA_{j'} \in \mathbb{F}^j_{\aleph}$ **then**
    select $S_{j,\circledast}$ with the minimum $J_j\,(S_j)$
    SEND $(S_{j,\circledast})$ to $CA_{j'} \in \mathbb{F}^j_{\aleph}, \forall i$
else if $CA_j \in \mathbb{F}^{j'}_{\aleph}$ and $CA_{j'} \in \mathbb{L}_{\aleph}$ **then**
    RECEIVE $\left(S^L_{j',\circledast}\right)$ from $CA_{j'}$ for which $CA_j \in \mathbb{F}^{j'}_{\aleph}$
    for $z_f = 1$ to LENGTH(S_j) **do**
        $\eta := S^L_{j',\circledast} - S_{j,z_f}$
        $J_j\left(S_{j,z_f}\right) := J_j\left(S_{j,z_f}\right) + \omega \cdot \mathbf{CompCnf}\,(\eta, G_N, \lambda, \mu)$
    end for
    select $S_{j,\circledast_f}$ with the minimum $J_j\,(S_j)$
end if
if $\uparrow G^d_{j,0} = t_0$ **then**
    switch the traffic light of I_j
end if
EndFunction

Line 13 from the Algorithm 5.4

$$J_j\left(S_{j,z_f}\right) := J_j\left(S_{j,z_f}\right) + \omega \cdot \mathbf{CompCnf}\left(\eta = S^L_{j',\circledast} - S_{j,z_f}, G_N, \lambda, \mu\right) \tag{5.19}$$

shows how a follower incorporates, biased by ω, the cost for not-following the leader's desired schedule. The weight of the not-following cost is ω. In our experiments we tested different values for ω to understand its influence over the performance of the leader and follower, respectively.

5.4 Validation examples

In our experiments we use five intersections connected in a star network topology as shown in Figure 5.1. Intersection I_1 is selected as leader and the rest of the intersections are its followers. Initially, the simulation starts with an empty network. Therefore, the optimisation only begins after a couple of cycles, such that the network is filled with vehicles. The optimisation starts at time 60 [sec.] such that the first optimisation is run just before the second possible switching at time 60[sec.]. The parameters used in the optimisation are minimum green phase $\underline{G_j} = 30$ [sec.], maximum green phase $\overline{G_j} = 60$ [sec.], and a nominal green period of $G_N = 32$ [sec.]. Each local controller uses an optimisation horizon $T_h = 130$ [sec.]. In order to validate our leader-follower framework, we use:

- no right or left turning traffic in the intersections,
- same driving speed on the roads connecting the intersections,
- same duration needed to cross the intersection,
- platoon size not changed in between intersections (there are no vehicles coming from/going to parking or unmeasured side-roads).

All the above items represent possible sources of uncertainty in our model that will be later introduced for more realistic examples. For our experiments, we use the same inflows generated using the following characteristics $\mathbb{E}\left[N_{A_m}\right] = 3[veh.]$ (integers uniformly distributed in $[1, K_{max} = 5]$) and $\mathbb{E}\left[G_{A_m}\right] = 4$ [sec.] (exponentially distributed). Both parameters, $\mathbb{E}\left[N_{A_m}\right]$ and $\mathbb{E}\left[G_{A_m}\right]$ are used in (3.6) to define the average

arrival platoon rates at the boundaries of the urban traffic network as introduced in Section 3.4.2.

Always repeating the same inflow from sources, no noise and only linked to the particular selection of the switching times, we compare our leader-follower solution with a fully distributed solution. The comparison is made for different values of $\omega = \{0,\ 1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ (defined in (5.19)). For $\omega = 0$, the fully distributed solution is equivalent with our approach since the not-following cost becomes zero. For $\omega = 1000$, the followers must do basically what their leader requests such that the local feedback at the follower does not count any more.

The performance of the network is assessed by comparing the average queue lengths at each intersection. What we want to show is that even by exchanging a limited amount of information, the performance of the leader, and in most of the cases of its followers will improve. The leader-follower concept induces the green wave phenomenon in which a series of traffic lights are coordinated to allow continuous traffic flow over several intersections for all competing directions, in an approximately optimised way. Using the described set-up, we first look at a Manhattan grid type network.

5.4.1 Manhattan grid type network

The Manhattan grid type network refers to the original design plan for the streets with equal length in Manhattan, New York City. An equal time delay $(= 38\ \text{[sec.]})$ between the intersections of the network leads to a simpler solution to create green waves.

In order to show how our optimisation framework can automatically recover from a unsynchronised situation, the initial conditions (green phases) of I_4 and I_5 are different than the other two followers I_2 and I_3. Figure 5.18 shows the phases of the traffic lights using the leader-follower framework for $\omega = 0$.

We note that increasing ω leads to a better cooperation between the leader and its followers and on average to smaller queues at the leader. Using the same length for the links of the network makes the phases of the followers to be fully aligned with respect to the leader's phase shifted with the delay of the links (or very close to that delay) as shown in Figure 5.19. Between time 60 [sec.] when the optimisation starts and time 300 [sec.], we observe a transient behaviour due to the initial phases desynchronisation of the followers. After 300 [sec.], the desynchronisation effects disappear thanks to our

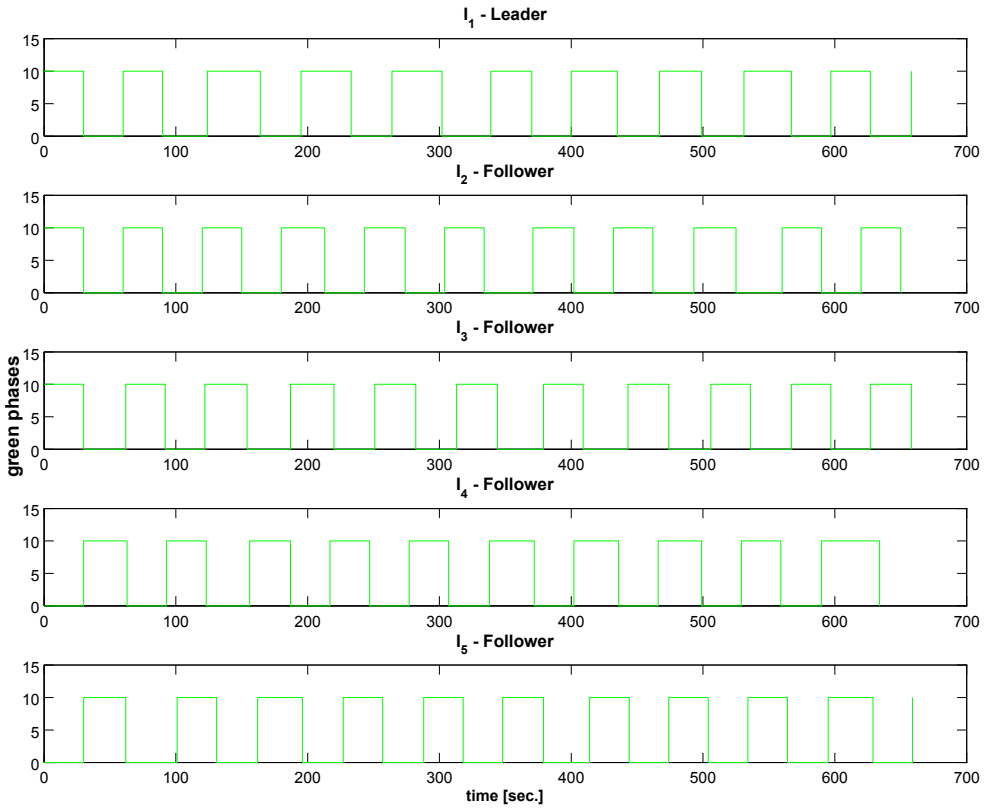


Figure 5.18: The phases of the 5 intersections using the leader-follower framework for $\omega = 0$.

leader-follower approach.

The average queue lengths (over the time interval 300 to 700 $[\text{sec.}]$ and for each ω) of the leader-follower framework at each intersection are compared with the ones obtained from the fully distributed approach as shown in Figure 5.20. More coordination between the leader and its followers leads to over two times smaller queues on average at the leader side and at worst an increase of 1 vehicle on average at the followers side compared with the fully decentralised case. For this particular input, the best performance is for $\omega = 1000$, when the queues at the leader (I_1) have a drop with approximately 64.75% from the value of the average queue lengths at the same intersection for the fully distributed approach. For the same $\omega = 1000$, the followers have a decrease between 1% and 12%. Sometimes a trade-off is made, thus an improvement of the leader brings a degradation of the performance of the followers (e.g. $\omega = 10$ or 100).

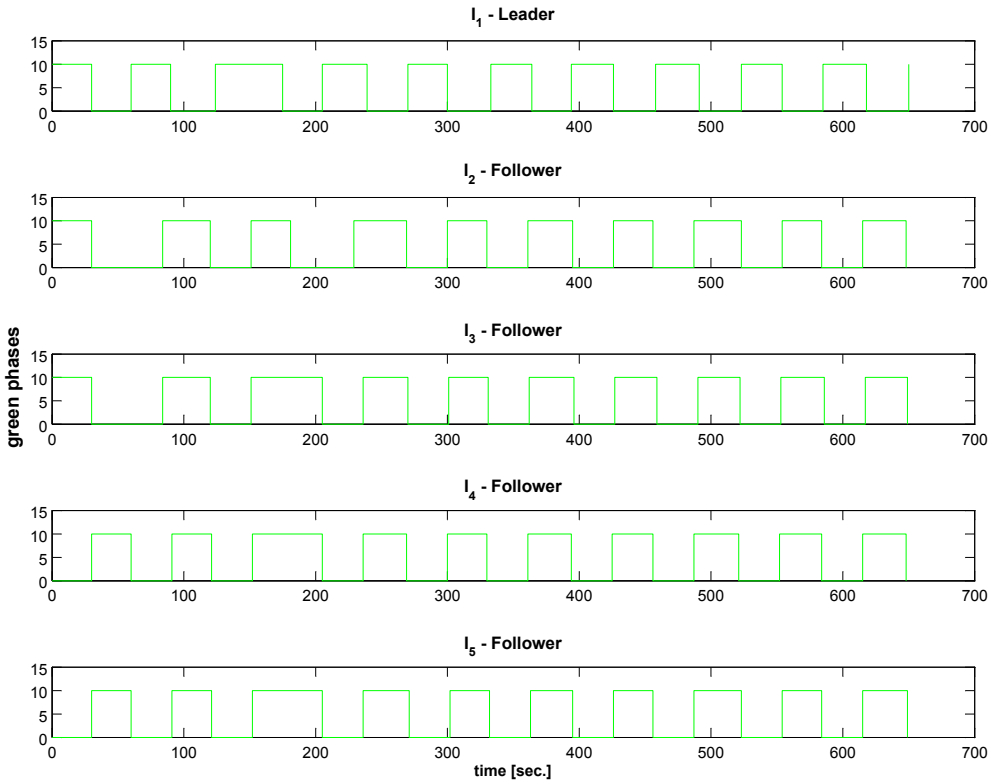


Figure 5.19: The phases of the 5 intersections using the leader-follower framework for $\omega = 100$.

For this first case, the optimal synchronised switching with just a fixed green wave is obvious thanks to equal link lengths of the Manhattan grid type network. For the other grid types that are typical for European urban traffic networks, the optimal green wave is not so clear anymore and factoring in local desiderata will become more meaningful. This last remark leads us to the next example for a Non-Manhattan grid type network.

5.4.2 Non-Manhattan grid type network

The validation of the leader-follower framework continues with the investigation of the same topology as in Figure 5.1, but using different lengths for the links connecting the intersections. For a Non-Manhattan network case, it is harder for the followers to keep up with the leader since each leader-follower pair has a different optimal green

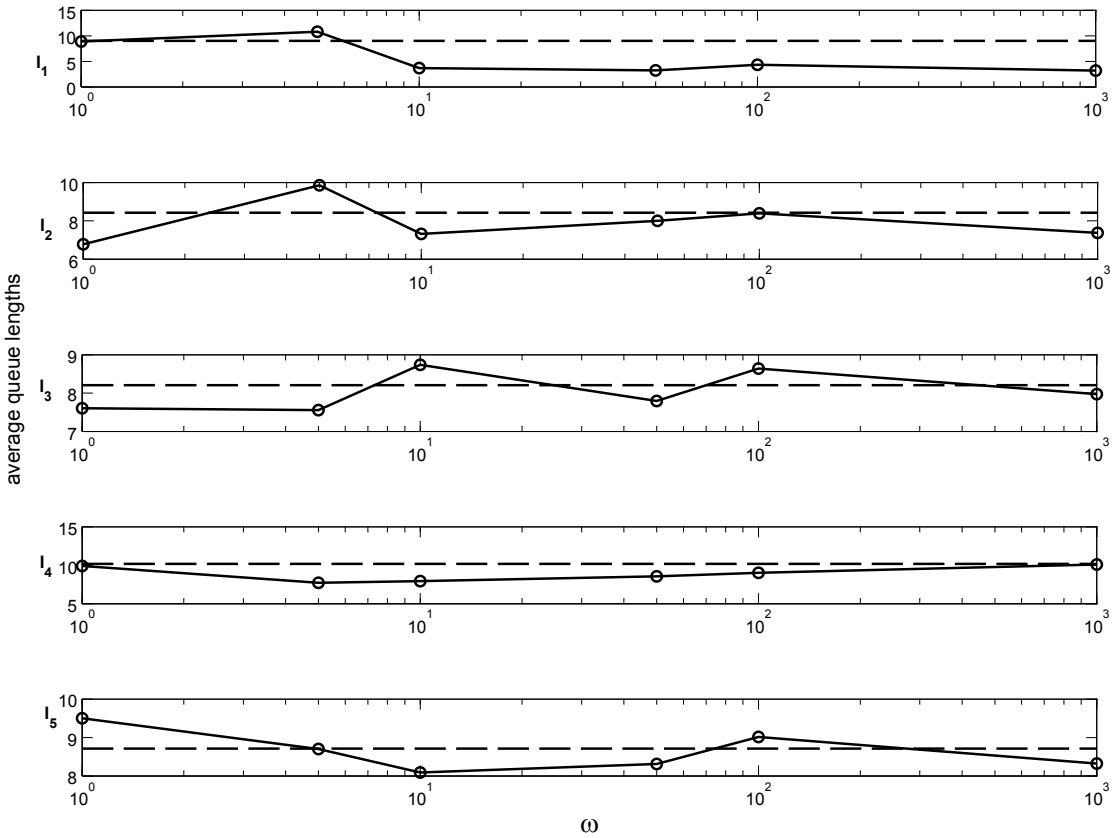


Figure 5.20: The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for Manhattan grid type network without noise. The performance of the fully distributed controller equivalent to $\omega = 0$ is represented in dashed line.

wave. For example, serving both East-West and West-East traffic for one input requires a time offset δ_l from East to West and from West to East, thus a cycle time of $2\delta_l$ at both intersections creates a green wave. When one intersection is the endpoint of two links with different offsets (e.g. δ_{l1}, and δ_{l2}), there is a conflict since the cycles times $2\delta_{l1}$ and δ_{l2} are different so an optimal green wave is harder to be found.

We consider the following link lengths: the North links - $300\,[m.]$, the East links - $600\,[m.]$, the South links - $450\,[m.]$, and the West links - $400\,[m.]$. We use the same $\omega = \{0,\ 1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ for our leader-follower framework and the results are presented in Figure 5.21.

For this particular input, the average queue lengths (over the time interval 300 to 700 [sec.] and for each ω) in the case of the leader-follower framework are in most of the cases better than the fully distributed solution. The best performance is achieved for $\omega = 5$, when the queues at the leader (I_1) have a drop with approximately 37% from the value of the average queue lengths at the same intersection for the fully distributed approach. The followers have a decrease of the values between 8% and 18% with the exception of I_5 that has a negligible improvement.

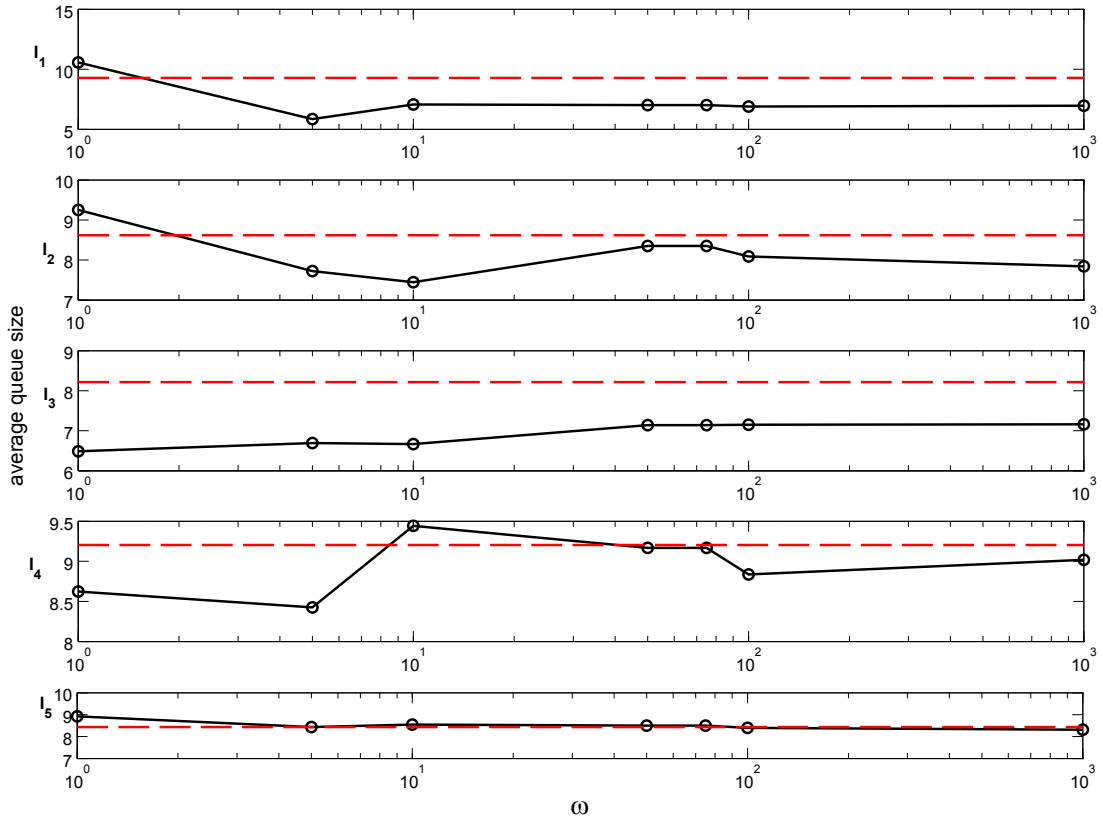


Figure 5.21: The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for Non-Manhattan grid type network without noise (in dashed line the performance of the fully distributed controller equivalent with $\omega = 0$).

5.5 More realistic examples

After we validated the newly introduced leader-follower framework by comparing it with a fully distributed set-up using the same inflows, we now investigate the robustness of the feedback control system. Sources of uncertainty in the network considered in the current section are: the driving speed on the roads connecting the intersections, noise along the link coming from the vehicles that leave or join a platoon (these are vehicles coming from/going to parking spaces or unmeasured side-roads), turning ratios for each direction of each intersection, and the duration needed to cross the intersections. In each case we mention the noise sources used in the simulations. Similar with the validation example, the optimisation starts at time 60[sec.] and uses $\underline{G_j} = 30\,[\text{sec.}]$, $\overline{G_j} = 60\,[\text{sec.}]$, a nominal green period of $G_N = 32\,[\text{sec.}]$, and an optimisation horizon of $T_h = 130\,[\text{sec.}]$.

We compare again the leader-follower framework for different values of ω with the fully distributed solution based on $\mathcal{N}_{\text{sim}}$ runs. Remember that for $\omega = 0$, the fully distributed solution is equivalent with our approach since the not-following cost becomes zero. For $\omega = 1000$, the followers must do basically what their leader requests such that the local feedback at the follower does not count any more. We use the same inflow for each run (identical sequence of platoons at the boundaries of the network for each ω and for the distributed approach) with the same characteristics as defined in the previous section ($\mathbb{E}\left[N_{A_m}\right] = 3[veh.]$ and $\mathbb{E}\left[G_{A_m}\right] = 4[sec.]$).

We use boxplots to present the results as a convenient way of graphically presenting groups of numerical data through their quartiles. The number of runs $\mathcal{N}_{\text{sim}}$ (its value is mentioned for each case) represents the number of repetitions based on which statistical results are presented. The quartiles of a ranked set of data values are the three points Q_i that divide the data set into four equal groups:

- Q_1 first quartile (splits off the lowest 25% of data from the highest 75%),
- Q_2 second quartile (cuts the data set in half),
- Q_3 third quartile (splits off the highest 25% of data from the lowest 75%),

where $Q_1,\ Q_3$ are represented by the bottom and top of the box, and the red line (the median of the data set) represents Q_2. The $\top,\ \perp$ symbols depict the highest and

lowest values excluding the outliers that are represented with + symbols. The values are drawn as outliers if they are larger than $Q_3 + 1.5 \cdot (Q_3 - Q_1)$ or smaller than $Q_1 - 1.5 \cdot (Q_3 - Q_1)$, where Q_1 and Q_3 are the first and third percentiles, respectively. This corresponds to approximately 99.3% coverage if the data are normally distributed. The last column labeled "all" from each figure represents the cumulative results from all five intersections (5 x $\mathcal{N}_{\text{sim}}$ runs) for the fully distributed set-up.

5.5.1 Manhattan grid type network with noise on the links

For the Manhattan grid type network with equal lengths ($= 600\,[m.]$) for the links connecting the intersections, we investigate if our framework can handle noise in the system. The noise sources considered for this example are:

- each platoon will move through the links with an independently chosen speed $v_n = p \cdot V_{max}$, where $p = 0.8, 0.9$ or 1 with respective probabilities $.03, .07, .09$ and maximum speed $V_{max} = 60\,[km/h]$,
- vehicles that can enter/leave on the links as in (3.5), where $N_{enter,i,n}$ and $N_{leave,i,n}$ are integer values drawn uniformly from $[-p_i \cdot N_{i,n} : p_i \cdot N_{i,n}] = [-1, 0, 1]$ (where $N_{i,n}$ is the number of vehicles in platoon $P_{i,n}$ passing location i, and p_i is the percentage of vehicles that can leave or join the platoon $P_{i,n}$),
- no vehicles turn, they all go straight at intersections.

The robustness of the leader-follower framework is tested through $\mathcal{N}_{\text{sim}} = 20$ simulations for each $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 500,\ 1000\}$ and presented using boxplots in Figure 5.22.

As can be observed, for $\omega = 500$, the average queues at the leader (i.e. I_1) decrease with approximately 30% compared to the fully distributed case, and at the followers with reductions between 3% and 10%. For $\omega = 1000$, the average queues at the leader are with approximately 50% smaller than the fully distributed case and for the followers with a decrease between 5% and 10% smaller.

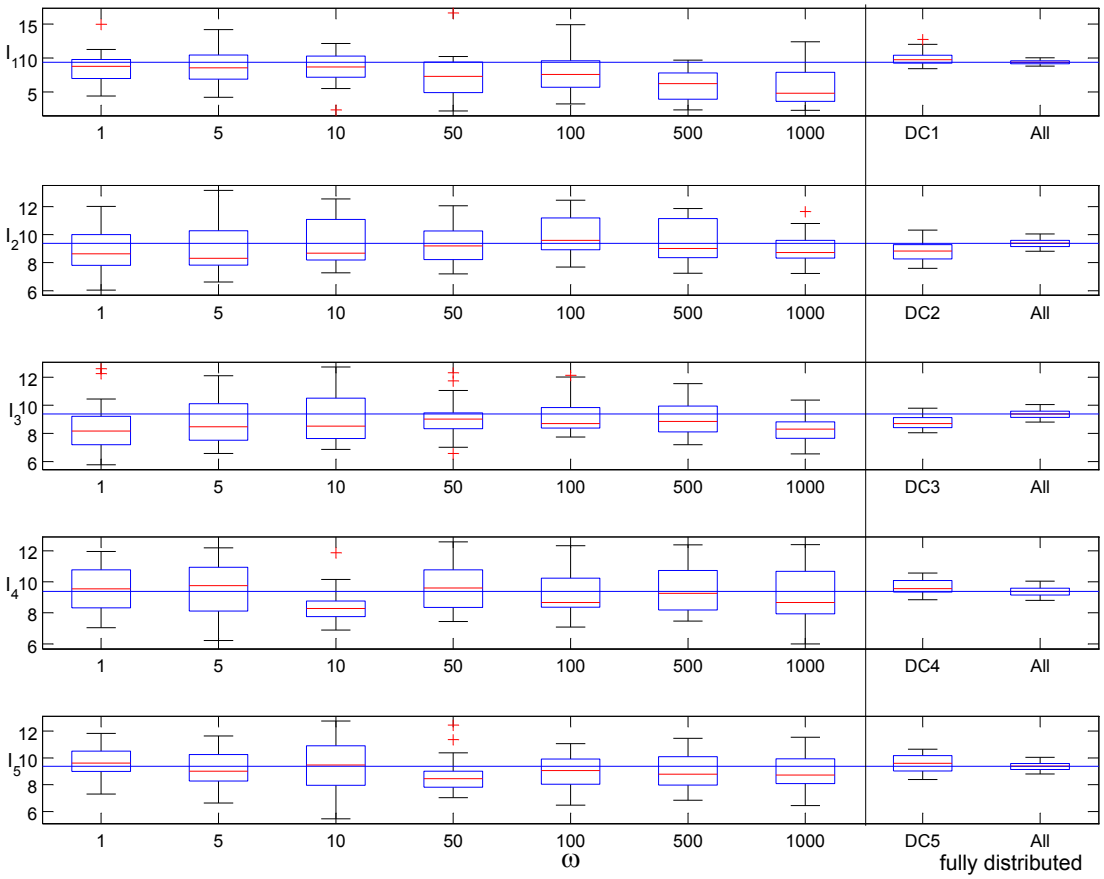


Figure 5.22: The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for a Manhattan grid type network with noise on the links (red lines represent the median values of the data set and the blue horizonal line that goes through the whole graph represents the median for the network using the fully distributed approach, whose value is also represented by the rightmost bar-plots).

5.5.2 Manhattan grid type network with right-turning traffic

So far the right-turning traffic was not used in our experiments. Without right-turning traffic, the leader has more control over the arrival of platoons (recommend switching red to its followers such that no platoon arrives). With right-turning enabled, the difference of the traffic output between the green and red phases at the follower intersections is reduced. Thus, we expect to have less control through the switching. As a consequence, it makes less difference (i.e. gives less improvement) to choose a particular "optimal" sequence. The turning ratios are defined for each input of each intersection of the

topology as in Figure 5.1 with probabilities .1, .9, 0 (i.e. no left turning traffic) for going *Right*, *Forward*, and *Left*, respectively.

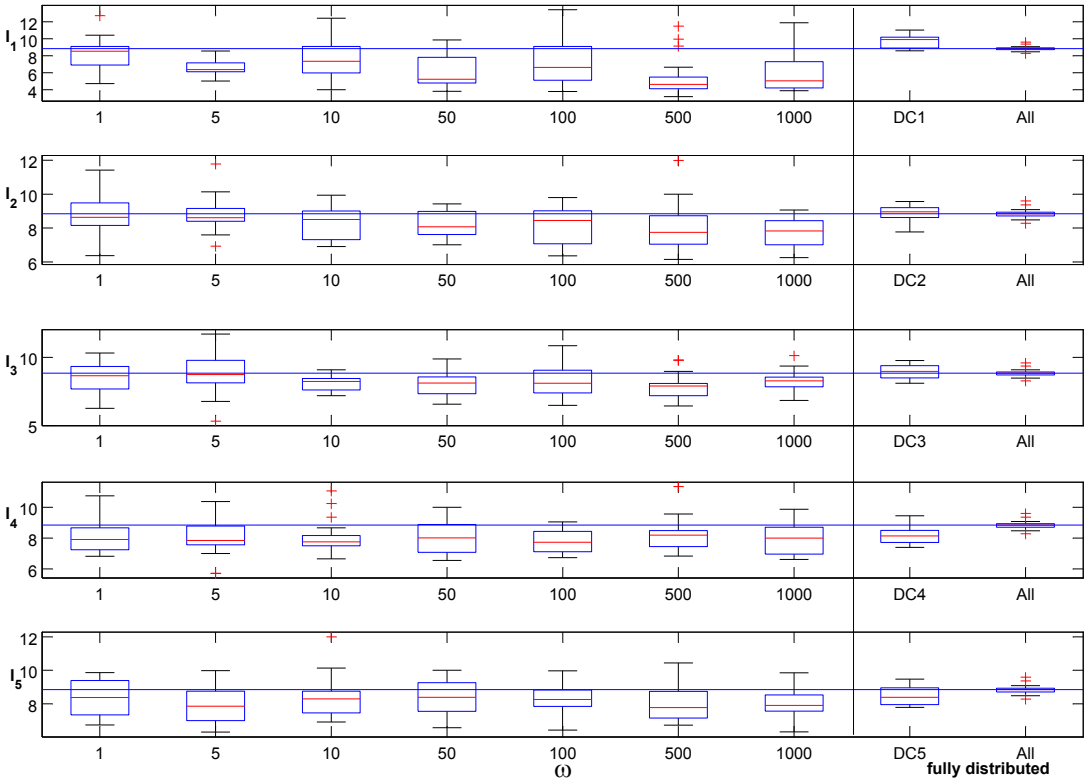


Figure 5.23: The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for a Manhattan grid type network with noise on the links (red lines represent the median values of the data set and the blue horizonal line that goes through the whole graph represents the median for the network using the fully distributed approach, whose value is also represented by the rightmost bar-plots) and right turning traffic.

We run $\mathcal{N}_{\text{sim}} = 30$ simulations for each $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 500,\ 1000\}$ and the results are presented using boxplots in Figure 5.23. In this case, for $\omega = 500$ and 1000, the average queues at the leader decrease with values between 40% and 50% compared to the fully distributed case, and for the followers with values between 5% and 15%.

However, here we could expect that the highest value of ω would be the best, because of the particularly simple properties of the Manhattan grid, where a clear optimal green

wave is easy to be found. In the case of more European-like urban networks, just fully synchronizing will not be optimal, so it makes sense to compute all cost options as a value of ω and one of these values could give better performance.

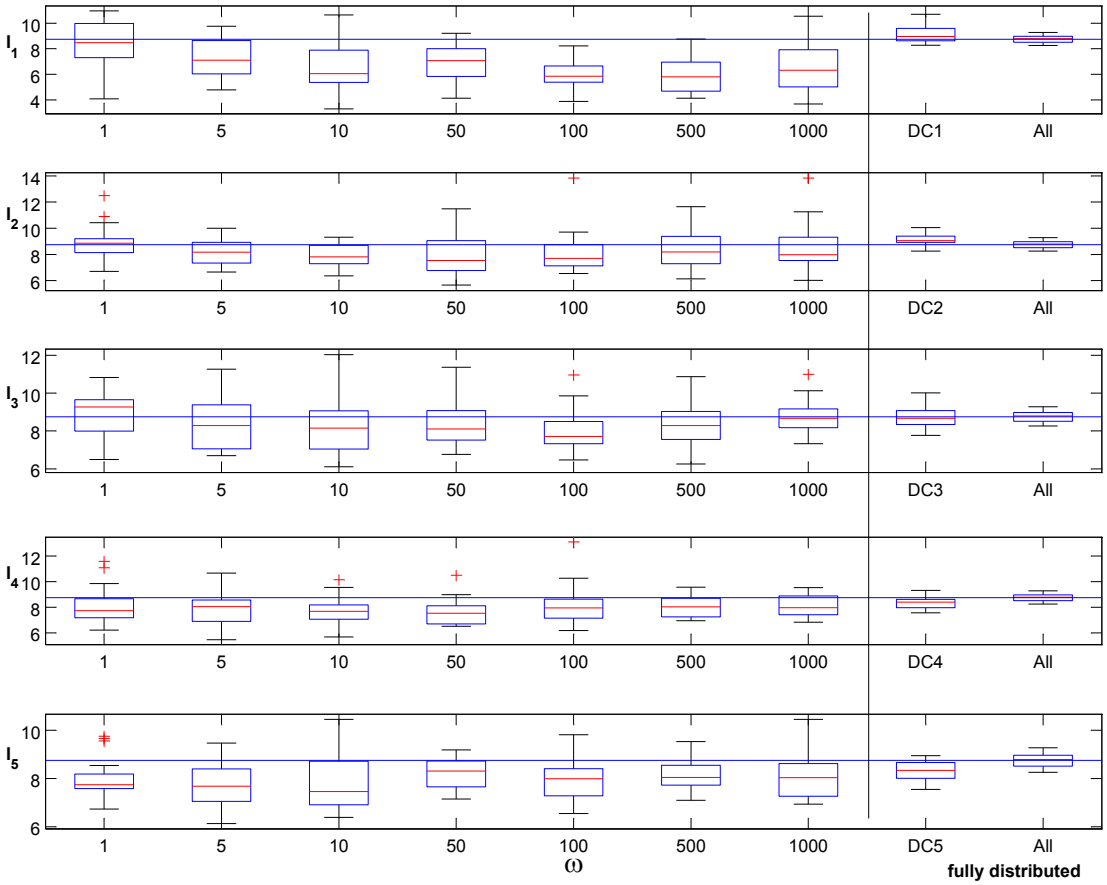


Figure 5.24: The average queue lengths for different $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 1000\}$ at the leader and its followers for a Non - Manhattan grid type network with noise on the links (red lines represent the median values of the data set and the blue horizonal line that goes through the whole graph represents the median for the network using the fully distributed approach, whose value is also represented by the rightmost bar-plots) and right turning traffic.

5.5.3 Non-Manhattan grid type network with right-turning traffic

The third example is the closest to reality since we not only introduce all types of noise, but also the topology of the network has different lengths for the links connecting the intersections same as in Section 5.4.2. We run $\mathcal{N}_{\text{sim}} = 20$ simulations for each $\omega = \{1,\ 5,\ 10,\ 50,\ 100,\ 500,\ 1000\}$ and the results are presented using boxplots in Figure 5.24. In this case, for $\omega = 500$, the average queues at the leader decrease with approximately 30%, and for the followers with values between 5% and 12%. Notice that for a stronger coordination between the leader and its followers (for $\omega = 1000$), the average queues at the leader decrease with approximately the same value as for $\omega = 500$, but the performance of the followers is closer to the fully distributed case and for I_3 the median is the same.

	$\%$ reduction in average queue length at				
	I_1	I_2	I_3	I_4	I_5
$\omega = 500$	35.24	9.66	4.37	4.41	3.52
$\omega = 1000$	29.42	11.98	0.09	5.11	3.64

Table 5.2: Reduction compared with the fully distributed approach (in percents) of the average queue lengths per each intersection for the highest two ω values when using the leader-follower framework.

For this example, with a Non-Manhattan grid with noise and for 20 runs, less coordination ($\omega = 100$) brings overall a better improvement. For $\omega = 100$ the average queues at the leader are with approximately 30% smaller and the followers have values of approximately 10% smaller. Table 5.2 presents the reduction (in percents) of the average queue lengths per each intersection for the highest two ω values when using the leader-follower framework compared with the fully distributed solution. The effects of a Non-Manhattan grid network (different optimal green wave for each leader-follower pair) and right-turning traffic (less control over the switching) sum-up and lead to an even more difficult case for finding and optimal green wave.

5.6 Summary

In the current chapter we have developed a new control framework for distributed control of urban traffic networks. We improve the performance of the distributed controllers by allowing coordination between the control agents through the exchange of relevant information, while maintaining flexibility in terms of control actions. Using switching times instead of the more traditional parameters (cycle length, split and offset) creates the possibility to react faster to the changes of traffic.

After a short literature survey, we introduced the theoretical aspects of the leader-follower framework. We continued by investigating different examples of real life situations for a network with 5 intersections connected in a star topology (leader provides a common connection point for 4 leaders like in Figure 5.1). In each example we show the superiority in performance of the leader-follower framework over a fully distributed control strategy.

Compared with a nominal green wave based, our approach creates a dynamic green wave that can adapt to the traffic variations. In Figure 5.24, it seems that with $\omega = 1000$ (closer to the nominal green wave - followers pay a very high cost for not following and the local optimisation is almost dropped) the set-up performs worse compared to the cases when using $\omega = 100$ or $\omega = 500$. Of course, more experiments have to be performed to ensure the validity of this remark.

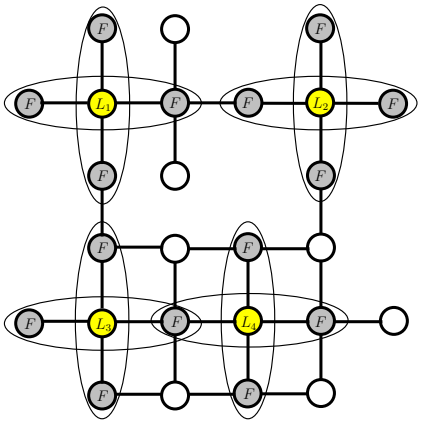


Figure 5.25: A network with 4 leader agents (coloured in yellow), 15 follower agents (coloured in silver, one follower has two leaders) and 7 agents (neither leaders or followers) that do not have measurements and apply some nominal green cycles.

Further experiments on larger networks including two or more leaders, followers with more leaders and agents that do not have measurements and apply some nominal green cycles. A configuration of more leaders at one follower can be easily realised by adding the multiple C_{nf} costs (corresponding to each leader) at the costs of all feasible scenarios from their respective follower. An example of such a network is presented in Figure 5.25.

Chapter 6

Modelling and distributed control of the autonomous vehicles in a container terminal

The chapter presents a case study for efficient automatic control of the autonomous vehicles (AVs): the supervisor that assigns tasks to the AVs, collision avoidance agents, and low level controllers (LLC) used to follow the trajectories assigned by the supervisor. The aim is to design a distributed collision avoidance control system that is robust with respect to communication failures, and that reduces as much as possible the intervention of the supervisor. In our work we assume that the position of any detected AV is known by each local agent up to a fixed error bound. We used world automata models, briefly introduced in Section 2.2.2, for analysing safety issues concerned with these interactions between the local agents and to help in designing the distributed collision avoidance control system for a number of autonomous vehicles, each of which executes tasks assigned by the supervisor. The content presented here was published in [52].

As joint work with the Electronic Systems Design group, University of Verona, our contributions are the system architecture, the control algorithm, and the implementation and validation of the model. Therefore, the focus here is not on presenting in more details the theoretical aspects of the World Automata framework. The interested reader can find a complete description of operators on WAs in [9] and references therein. The problem specifications were indicated by Jan Tijmen Udding in the setting of the

CON4COORD - EU FP7 project.

A similar problem is presented in [28], where the controller algorithm is described as a state-machine and the navigation algorithm for each AV as an extended Kalman filter (EKF). The implemented system suffers from a relatively high number of deadlocks that occur during normal operation. The issues of safety and provable systems performance are proposed as a major future research direction. In the literature, other collision avoidance solutions are available for different specific types of vehicles. In [7] a mixed-integer nonlinear model used with a global optimisation for the problem of Unmanned Aerial Vehicles (UAVs) conflict resolution is presented. Conflicts are avoided allowing the UAVs to only accelerate or decelerate in a time window, and speed changes are minimised together with time windows when they occur. We use the same principle of decelerating for avoiding the conflicts and, in addition, since we use autonomous ground vehicles, we also allow full stops. A decentralised receding horizon motion planner algorithm for AVs, based on artificial potential fields and sliding mode control technique, is presented in [23]. The artificial potential fields approach is difficult to be applied in our case (the yard layout: a long and narrow space). A more general treatment for the reciprocal n-body Collision Avoidance algorithm is presented in [76]. Compared with our case the AVs cannot communicate at all, and the AV model ignores kinematics and dynamics. In [67], the decentralised collision avoidance problem turns to be a challenging liveness-verification problem for a complex hybrid automaton and a probabilistic verification method is developed.

The system is carefully detailed in Section 6.1 continued by the presentation of the control problem in Section 6.2. The world automata framework is briefly reviewed in Section 2.2.2. The model of the system using the world automata framework is presented in Section 6.3. After the implementation of the model, we perform the system validation and present the results in Section 6.4.

6.1 Description of the system

Moving containers between trains, trucks, ships, and storage yards is one of the most important operations in the harbours of our days. The essence stands in moving an increasingly large number of containers in the shortest time with the highest safety. One type of vehicles used for stacking and moving containers in port terminals and

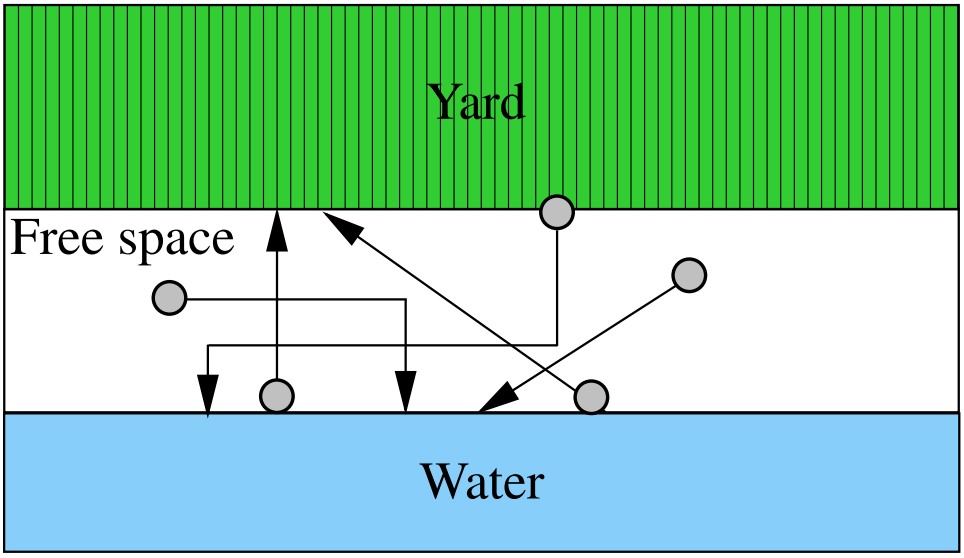


Figure 6.1: Schematic representation of the port quay of Antwerp. The stripes represent the row of containers, and the circles represent the straddle carriers. The row of cranes placed along the quay on the water side is not represented.

container yards is the straddle carrier (ST) (here called autonomous vehicle). The human driven STs pick up, transport, and stack containers vertically up to 3 high (with 60 tons max.load/container). Figure 6.1 shows the yard layout of a harbour, with the row of cranes near the water, the quay where the STs can move freely and the stacks of containers (where STs must follow the lanes between stacked containers).

We propose an automated system of the operations carried out by STs without drivers, autonomously carrying out instructions to pick up containers and transport them to their destination, taking in consideration the safeness of the system. The automated system is intended to work in the free space area presented in Figure 6.1.

6.1.1 System architecture

The architecture of the system contains a supervisory layer (SL) or supervisor that assigns tasks to the N autonomous agents CA_i autonomously controlling vehicles $AV_i,\ i = 1, \ldots, N$, as in Figure 6.2. We need to make a clear distinction between the control layers hierarchy presented here and the modelling hierarchy defined later in Section 6.3.

Definition 6.1. *The task Tsk$_n$ is defined by its reference trajectory*

$$\textit{Ref}_n^a : [T_n^1, T_n^{2,a}] \to S,\ T_n^1 < T_n^{2,a} \leq t_{D_n}, \tag{6.1}$$

with the following elements:

- a *represents the number of update of the reference trajectory to avoid collisions (SL assigns initially* Ref_n^0*),*
- $T_n^1, T_n^{2,a}$ *start and end time of the task,*
- *deadline* t_{D_n} *when task must be completed,*
- *initial position* $\textit{Ref}_n^a(T_n^1) = O_n$ *and final position* $\textit{Ref}_n^a(T_n^{2,a}) = D_n$ *both located on the boundary of* S*.*

where the reference trajectory is a polygonal line connecting O_n *to* D_n

Each AV_n is assigned by the SL to perform Tsk$_n$, $n \in \mathbb{N}$, autonomously solving the possible conflicts (except in emergency cases). The tasks are assigned such that the AVs move in a rectangular workspace (yard) $S \subset \mathbb{R}^2$.

The supervisor selects the initial time T_n^1 such that

- the vehicle can satisfy the deadline t_{D_n},
- the risk of collisions with other vehicles is reduced, if possible, based on coarse estimates of the space occupied by the other vehicles.

Definition 6.2. *The path* $\gamma_n(t)$ *of* AV_n *at time* t*, executing Tsk$_n$ is the set of points in space to be visited by* AV_n *after time* t*. To each reference trajectory* $\textit{Ref}_n^a$ *there corresponds a path* $\gamma_n(t) \subset \mathbb{R}^2$*, defined as the set of points* $\textit{Ref}_n^a(\tau), \forall \tau \in [t, T_n^{2,a}]$*. The default velocity of the vehicle along this path is the maximum speed* v_{max}*, identical for all vehicles.*

The allocation of the trajectories determined by the supervisor is not necessarily free of collisions with other vehicles. Since a path can not be guaranteed as collision free by the supervisor, the CA$_n$ of AV_n must from time to time locally decide to modify its trajectory Ref_n^a to $\text{Ref}_n^{a+1} \in S$ if it detects a risk of collision. Therefore, the goal of the collision avoidance algorithm is to solve the eventual risk of collisions and to make each AV to fulfil its task with a minimum time delay, completing the task by t_{D_n}.

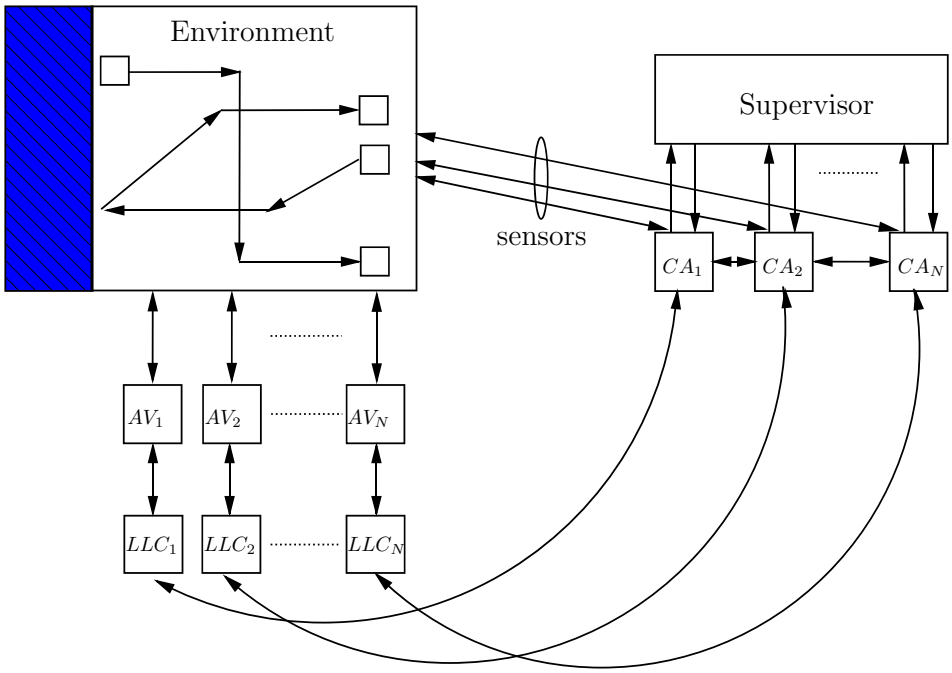


Figure 6.2: Architecture of the system. Note that the controllers are physically located in AVs (e.g. CA_N in AV_N).

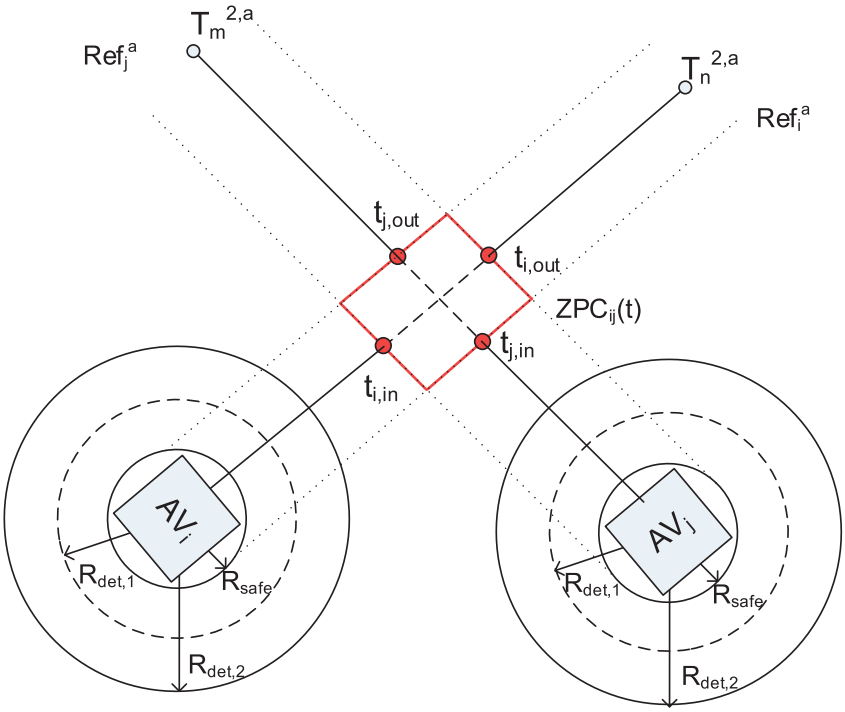


Figure 6.3: AV_i in a possible collision with AV_j

Definition 6.3. *The* margin *of* AV_n *at time* $\tau \in [t, t_{D_n}]$ *is* $M_n(\tau) = t_{D_n} - T_n^{2,a}$ *as the time reserve of* Tsk_n *to be executed (with* $T_n^{2,a} < t_{D_n}$*).*

If any AV_n is not able to meet its given deadline t_{D_n} (which can be determined by CA_n on the basis of $\gamma_n(\tau)$ and $M_n(\tau)$ at time $\tau < t_{D_n}$), the supervisor is informed immediately (at time τ) by the LLC of AV_n that the task could not be fulfilled within the given time and with the given trajectory. Hence a new trajectory is assigned by the supervisor with a new deadline.

It is clear that an increase in the number of the vehicles will lead to an increase of this collision avoidance manoeuvres and thus an increase of the average delay. We assume that a permanent communication channel exists between the SL and AVs but using is expensive. The AVs can also try to communicate between each other using an unreliable communication channel in a collision avoidance case. First, the AVs try to establish communication and to solve the conflict using the exchanged information. If the communication attempt fails, the collision avoidance algorithm shall be able to

make a decision only based on its own local measurements in order to make the system robust against communication failures. AVs can use the communication with the SL only as a last resort (if the collision avoidance algorithm fails) since using is expensive.

6.1.2 Autonomous vehicle specifications

Any AV_n is equipped with:

- an unique identification tag ID_n,
- a sensor makes noisy detections about the other AVs neighbouring approximate directions and distance to obstacles around AV_n as follows
  - within a radius R_{safe} (the inner-most circles around an AV_n) the AV_n fully stops when obstacles are detected,
  - within a radius $R_{\text{det,1}} > R_{\text{safe}}$ all obstacles are detected with certainty,
  - obstacles within $[R_{\text{det,1}}, R_{\text{det,2}}]$ are detected with probability p (< 1),
  - obstacles at distance $> R_{\text{det,2}}$ are not detected,
  - the detection noise is uniformly distributed random variable on the open interval (D^{min}, D^{max}) added to the real position of the detected AV,
- since the yard is a rough environment (with lots of metal structures, causing Faraday cage effects), the wireless communications link used to exchange messages with other AVs that are within a range less than $R_{\text{det,2}}$ of AV_n is unreliable,
- a local control agent CA_n running the collision avoidance algorithm for AV_n,
- a lower level controller (LLC) that implements a control algorithm adjusting speed and direction, so as to follow Ref_n^a with bounded error (both speed and direction selected from a finite set of possible values).

To summarise, the relation between the radii of AV_n is

$$R_{\text{det,2}} > R_{\text{det,1}} > R_{\text{safe}}, \tag{6.2}$$

where every AV within R_{safe} is considered to be in collision with AV_n and every AV outside $R_{\text{det,2}}$ can not be detected by AV_n.

Definition 6.4. *The simplified dynamics of AV_n using the kinematic model are*

$$\dot{p_n}(t) = \overrightarrow{v}_n(t), \tag{6.3}$$

where $\overrightarrow{v}_n(t)$ is the controllable velocity (with the direction of $\overrightarrow{v}_n(t)$ tangent to $\gamma_n(t)$) and $p_n(t) \in S$ is the position of AV_n at time t.

The AV model is further simplified by discretising the speed of AV_n. This is partly in order to also simplify the estimation problem.

Definition 6.5. *The set of allowed input speeds of AV_n requires that $\|\overrightarrow{v}_n(t)\|$ belongs to the finite set:*

$$V := \{v_n = zv_s, z = 0, 1, \ldots, M\}, \text{ i.e. } \|\overrightarrow{v}_n(\tau)\| \in V \tag{6.4}$$

where $\tau \in [T_n^1, T_n^{2,a}]$, and v_s is the constant speed increment step for simplifying the speed estimation.

In practice, Definition 6.4 simplifies the speed estimates computation. Every reference trajectory is designed by the supervisor such that the corresponding vehicle moves at a velocity $v_{max} := Mv_s$, except during collision avoidance.

Definition 6.6. *The safety area $SA_n(t) = \{s : \|s - p_n(t)\| \leq R_{safe}\}$ around AV_n is the disk with radius R_{safe} and centre $p_n(t)$.*

In the example from Figure 6.3, AV_i and AV_j have to execute the missions Tsk_i, and Tsk_j, respectively.

Definition 6.7. *We define a **Z**one of **P**ossible **C**onflict (ZPC) at time t between two vehicles AV_i and AV_j (red polygon in Figure 6.3) as*

$$ZPC_{ij}(t) = (\cup_{\tau\in\Delta_i} SA_i(\tau)) \cap (\cup_{\tau\in\Delta_j} SA_j(\tau)),$$
$$p_i(\tau) = \mathit{Ref}_n^a(\tau), \forall\tau \in \Delta_n := [t, T_n^{2,a}], n = i, j. \tag{6.5}$$

The entry time $t_{n,in}$ into the ZPC and the exit time $t_{n,ex}$ out of the ZPC of vehicle $n = i, j$ at time t are defined as

$$t_{n,in}(t) := \min\{\tau | p_n(\tau) \in ZPC_{ij}(t)\}, \tag{6.6}$$

$$t_{n,ex}(t) := \max\{\tau | p_n(\tau) \in ZPC_{ij}(t), \tau \leq T_n^{2,a}\}. \tag{6.7}$$

Definition 6.8. *The **S**ufficient **C**ondition **E**xcluding **C**ollisions (SCEC) at time t between any two vehicles i and j, denoted as $SCEC_{ij}(t)$ is defined by*

$$t_{i,in}(t) < t_{j,ex}(t) \text{ or } t_{j,in}(t) < t_{i,ex}(t). \tag{6.8}$$

Definition 6.9. *The conflict group (CG) at a time instant t that corresponds to AV_i is:*

$$CG_i(t) := \{AV_j | SCEC_{ij}(t) \text{ is not satisfied}\}. \tag{6.9}$$

The SCEC is only a sufficient condition for the absence of collisions. If the SCEC is not satisfied, a collision may occur. When a SCEC does not hold for a pair of vehicles, we say that a conflict is detected. This introduces an extra safety margin and makes the algorithm more robust. Whenever a conflict is detected at time t by AV_i then CA_i will decide whether or not to adjust the reference trajectory Ref_n^a to a new trajectory Ref_n^{a+1}, depending on its priority in its conflict group (see 6.2.1 and 6.2.2 for the determination of priority). The failure of the collision avoidance algorithm leads to a deadlock.

Definition 6.10. *A deadlock occurs when two or more AVs within the same conflict group CG are stopped ($\|\overrightarrow{v}_n(t)\| = 0$) without the possibility to perform the assigned tasks.*

Notice that the vehicles are not required to work with synchronous clocks: every time step of Δ_i, AV_i senses its environment and performs a corresponding control action.

6.1.3 Controller specifications

Since the controller CA_i of each AV_i possesses only noisy measurements of its immediate surroundings, the collision avoidance algorithm can only apply a myopic, but cooperative action (based on limited information). This action will typically result in an increase of $T_i^{2,a}$. It is desirable for the algorithm of the controller of AV_i to introduce minimal extra delays. However, the local actions of AV_i at time t may influence both the conflicts AV_i will be involved in at future times, as well as the conflicts that may occur between other AVs. If a local controller fails to respect the deadline of the assigned task, it will send a message to SL. The SL will then reschedule/adjust the tasks of some vehicles in order to achieve the global goals. The rescheduling of tasks is outside the scope of the current work. The specifications for the problem thus are as follows

- **hard specification:** no two AVs should at any time come closer to each other than $2R_{\text{safe}}$ (to ensure that a collision is not possible) i.e. $\forall\ i, j, t, |p_i(t) - p_j(t)| > 2R_{\text{safe}}$.
- **soft specification:** the reference trajectories should be selected such that after a finite number a of conflict avoidance actions, the task gets completed by AV_i reaching $\text{Ref}_n^a(T_n^{2,a}) = D_n$ at time $T_n^{2,a} \leq t_{D_n}$. If exceptional circumstances make it impossible to achieve this specification, then the AV_i must send a message informing the supervisor of this violation of the soft specification. The SL sends new instructions to all currently active AVs.

The performance of the systems can be enhanced by the wireless communication system that allows the AVs that are within the detection range ($< R_{\text{det},2}$) of their sensors to exchange messages about their priorities. Since the wireless communication is unreliable, however the control algorithm shall be able to cope with both cases with and without communication, respectively.

6.2 Control problem

At time $k\Delta_i,\ k = 1, 2, \ldots$ the sensor of AV_i detects the approximated position $\hat{p}_j(k\Delta_i)$ and the estimated velocity $\hat{\vec{v}}_j(k\Delta_i)$ of the other vehicles within the range $R_{det,2}$. Although the AVs are modelled using simple integrators (as in Definition 6.4), the

interactions between them can be complicated. Each AV has its own clock that leads to an asynchronous detection of its neighbours ($\Delta_i \neq \Delta_j$). In order to represent the asynchronous interactions and limit their effect on the distributed control problem, we used the world automata framework.

Based on these measurements, AV_i constructs the set $S^k_{d,i}$ of detected AVs. After each update of $S^k_{d,i}$, AV_i tries to establish communication with each detected $AV_j \in S^k_{d,i}$ and two situations can appear:

- The communication between AV_i and AV_j is considered successful after the on exchange of acknowledgements indicating that the exchange of the reference trajectories (Ref^a_i and Ref^a_j) and margins ($M_i(k\Delta_i)$ and $M_j(k\Delta_i)$) was successful. If this case occurs, AV_i adds AV_j to the set $S^k_{r,i}$ (with $S^k_{r,i} \subseteq S^k_{d,i}$).

- In the case when the communication is not successful, the AV_i makes an estimate $\tilde{\text{Ref}}^a_j$ of the reference trajectory of AV_j and adds $AV_j \in S^k_{nr,i}$ (where $S^k_{nr,i} = S^k_{d,i}/S^k_{r,i}$). The AV_j's estimated trajectory is computed by assuming that it moves along a straight line and taking as inputs its recently estimated position $\hat{p}_j\left(k\Delta_i\right)$ and estimated velocity $\hat{\overrightarrow{v}}_j(k\Delta_i)$. This assumption is not dangerous since the AVs will always stop ($\|\overrightarrow{v}\| = 0$) if $|p_i\left(k\Delta_i\right) - \hat{p}_j\left(k\Delta_i\right)| < 2R_{\text{safe}}$.

  A deadlock, as in Definition 6.10, can appear in this case due to the uncertainty introduced by the noisy measurements. Therefore, the deadlocked AVs can solve the conflict through the permanent communication channel with the supervisor. Based on the assigned tasks and the time margins, the supervisor restarts the AVs (formally implementing and testing the supervisory layer is left for future work).

When the sets $S^k_{d,i}$ and $S^k_{r,i}$ have been updated, AV_i is ready to determine the presence of a possible conflict with the detected AVs. It checks for each $AV_j \in S^k_{d,i}$ if the $SCEC_{i,j}$ condition, defined by (6.8), is satisfied and builds the corresponding $CG_i(k\Delta_i)$. At the end of the check, AV_i creates $S^k_{c,i} = CG_i(k\Delta_i)$ the set of vehicles AV_j in conflict with AV_i.

At this point, any $AV_j \in S^k_{c,i}$ is already detected by AV_i i.e. $AV_j \in S^k_{d,i}$ since it can not be included in $S^k_{c,i}$ undetected by AV_i. Imagine a situation where AV_l is "shadowed" by AV_j (e.g. moving in parallel with approximately the same velocity) then AV_l is not detected by AV_i i.e. $\notin S^k_{d,i}$, nor in conflict with AV_i i.e $\notin S^k_{c,i}$.

The CA_i of AV_i autonomously makes a decision about changing or not Ref_i^a at time $k\Delta_i$, based on the following information:

- the current state of AV_i i.e. $p_i(k\Delta_i)$, $\dot{p}_i(k\Delta_i)$, Ref_i^a,
- the observations about its immediate surroundings, i.e. $AV_j \in S_{d,i}^k$ if $|p_i(k\Delta_i) - \hat{p}_j(k\Delta_i)| \leq R_{det,2}$ with $\hat{p}_j(k\Delta_i)$ the measured position of AV_j at time $k\Delta_i$,
- and the sets $S_{r,i}^k$, $S_{nr,i}^k$, $\tilde{\text{Ref}}_j^a$ (if the communication is not successful), $S_{c,i}^k$,

such that the closest conflict in time from $S_{c,i}^k$ is solved. A conflict clearly determines a reflexive relation between two vehicles: AV_i is in conflict with AV_j if and only if AV_j is in conflict with AV_i. Unfortunately, the way conflicts are *observed* by the vehicles is not reflexive: if AV_i observes a conflict with AV_j, it does not necessarily mean that AV_j observes a conflict with AV_i. This is due to:

- the asynchronous operation mode of the system: vehicles do not update their sensor readings simultaneously;
- $|p_i(k\Delta_i) - \hat{p}_j(k\Delta_i)| \in [R_{det,1}, R_{det,2}]$ representing the unreliability of sensors. In the case of achieved communication the estimate $\hat{p}_j(k\Delta_i)$ is replaced by the true value $p_j(k\Delta_i)$ sent by AV_j.

Based on the information, AV_i determines its priority in its conflict group. How the priority is determined depends on whether the communication was successful or not.

6.2.1 Successful communication case

In the case when the communication could be established, we can overcome the disadvantage of asynchronous operation mode of the system and to resolve the asynchronous detection problem. In this case both vehicles possess the same information about each other. Using the exchanged information, the AV with the smallest margin has the highest priority i.e. if $M_i(k\Delta_i) > M_j(k\Delta_i)$ then priority of AV_i is low else priority of AV_i is high. Assuming that their respective algorithms are identical, then both will reach the same conclusion about priorities. The rule_{pr} function returns the priority of AV_i based on the exchanged information with AV_j.

6.2.2 Unsuccessful communication case

The difficulty is in how to determine the priority in the case when the communication failed. In this case, the check of the corresponding $SCEC$ condition is based on the estimate of the reference trajectory of detected AV. Based on that estimate, the priority is determined using a list of rules. The priority rules are meant to be compiled in a nested if-then-else statements list with the purpose to break the symmetry of the actions taken by the AVs involved in the conflict. The possible outcomes of these rules are $\{0, 1, \text{undecided}\}$. The *undecided* outcome occurs when the inaccuracy of the measurements makes it impossible to decide e.g. which vehicle comes from the right. The priority rules that AV_i checks are:

1. "first in first out" - if the estimated difference $\|t_{i,in}(k\Delta_i) - \tilde{t}_{j,in}(k\Delta_i)\|$ is sufficiently large then the AV with the smallest entry time in the $ZPC_{i,j}(k\Delta_i)$ has higher priority,

2. "right hand priority" - if the angle between the paths of the vehicles is larger than the uncertainty margin then the AV coming from the right has a higher priority,

3. "loading containers priority" - the AV going towards the quay has a higher priority than the AV going towards the stack of containers,

4. "WestEast priority" - the AV coming from West has a higher priority than the one coming from East.

5. "toss-a-coin priority" - decide to give priority by tossing a coin.

The first four rules are only applicable if the resulting answer is unambiguous, which depends on the accuracy of the sensors. For example, the "right hand priority" depends on the measuring accuracy of the direction of motion of each conflicting AV (their angles must differ by more than the uncertainty on the measurement).

The third rule is justified by the necessity of giving priority to the loading operations (of a ship berthed at the quay). The last rule is the *toss-a-coin-algorithm* that will make a random choice of the priority. It is necessary to break the symmetry in assigning the priorities in the case where all the previous rules fail. The rule is not dangerous since the AVs come to a full stop ($\|\overrightarrow{v}\| = 0$) if the distance between them is $< 2R_{\text{safe}}$.

Furthermore, this rule simply reduces by half the number of times when the supervisor has to be invoked. The rule$_{pn}$ function returns the priority of AV_i according to the defined rules for the unsuccessful communication case.

From now on, let the vehicle with the highest priority be denoted by AV_{high}, and the other by AV_{low}. To resolve the conflict with a minimum delay, AV_{low} alters its trajectory, while AV_{high} keeps following its original reference trajectory. Although here we exemplify cases with only two AVs, the same manoeuvres and algorithms can be used for more than two. The AV_{low} has two options to alter its a-th trajectory which will be explained in the following subsection.

6.2.3 Manoeuvres

The a-th trajectory of AV_{low} is modified by either slowing down while maintaining the same trajectory or changing the trajectory $\text{Ref}^{\,a}_{n,\text{low}}$ by adding new vertices. The vertices are the corner points of the polygonal line describing the original reference trajectory $\text{Ref}^{\,a}_{n,\text{low}}$, making a new reference trajectory $\text{Ref}^{\,a+1}_{n,\text{low}}$.

Adjusting velocity while maintaining the path

To resolve the conflict, AV_{low} slows down to the highest speed that still ensures AV_{high} can pass without slowing down and without causing $SCEC_{high,low} = false$. The delay introduced by the "adjusting velocity while maintaining the path" manoeuvre is T_{as}.

Adjusting the path by adding vertices

Consider the reference trajectory $\text{Ref}^{\,a}_{n}$, $AV_i = AV_{\text{low}}$, which corresponds to a polygonal line. Call the current location $p_i(t)$ at time t and the destination of the trajectory *the outer vertices* of the polygonal line. The remaining vertices are called *the inner vertices*. If the path of AV_{low} has no inner vertices, we will adjust its path by adding one inner vertex in order to resolve the conflict. The T_{ar} is the time delay introduced by the "adjusting the path by adding vertices" manoeuvre.

If its path contains one or more inner vertices, we will move the vertex to be encountered first to a new position, yielding a new, collision-free reference trajectory. The optimal position of this inner vertex is obtained by calculating the time delay of

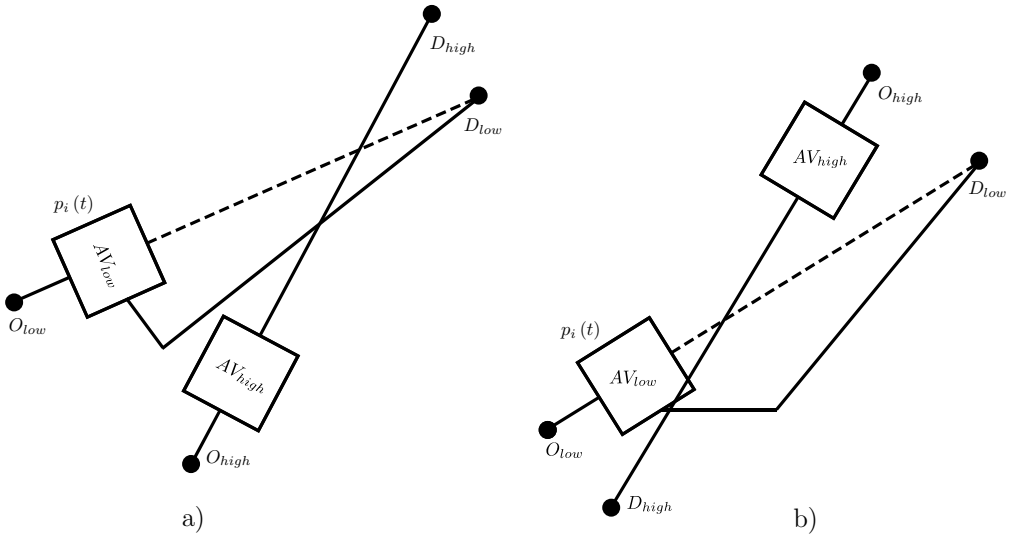


Figure 6.4: Adjusting the path by adding vertices for two possible scenarios (dashed line is the current reference trajectory and full line is the new adjusted path).

the new trajectory for all possible positions of this vertex inside a predefined area A around one or both vehicles AV_{high} and AV_{low}. As a rule of thumb, we theoretically consider all points of the workspace S that are located within a predefined distance R_{det} of the path of AV_{high} whereas a limited number of possible points is considered in practice. Intuitively, it is clear that in the absence of additional conflicts this problem will have at most two local minima, corresponding respectively with AV_{low} passing in front of AV_{high} or passing behind it. In some situations one or both of these minima may be prohibited due to restrictions on the space configuration, or conflicts with other AVs. Figure 6.4 shows the result of an optimisation procedure corresponding to the initial trajectory depicted in dashed line. The Figure 6.4.a shows that in this case AV_{low} passing behind the AV_{high} is the optimal solution. The time delay with respect to the initial trajectory is easily computed since the velocity along both initial and optimal trajectory are assumed constant and identical. In the same Figure 6.4.b shows how adjusting the path by adding vertices can be used to solve a possible head-on collision scenario.

The AV_{low} computes the respective delays T_{as} and T_{ar} corresponding to these manoeuvres (“adjusting velocity while maintaining the path“ and “adjusting the path by adding vertices“) and chooses the control action with the smallest delay.

In the case of unsuccessful communication only the “adjusting velocity while maintaining the path“ manoeuvre is used.

6.3 Model of the case study

We introduce here the representation of the system described in Section 6.1 with world automata. To preserve the modularity of the system, the AV_i model (presented in Figure 6.5) is separated from the estimator (in Figure 6.7) and the control agent CA_i (Figure 6.8). Since our focus is on the control agent CA_i the LLC_i module is incorporated in the AV_i model. Hence it is possible to change the controller and verify structural properties of the composed system with different control policies. Basically, estimator and control agent are HIOA, because they do not have any world variables. We represent them with the same pseudo-language instead of a graphical way to keep notation consistent with world automata theory.

6.3.1 Straddle carrier

An autonomous vehicle AV_i is modelled by the automaton of Figure 6.5. Several replicas of this model can be used to represent a scenario with many AVs, each one having its own identification tag $ID = i$. Each vehicle receives its reference trajectory (Ref_i^a) in the form described by (6.1). The AV moves according to (6.10), where $p_i(t)$ is the position of the AV and $u(t)$ is a vector representing the velocity (intensity and direction) given by the controller at time t.

Now, we explain how the communication failure, mentioned in Section 6.1, is modelled. The AV_i sends a message m within a communication delay Δ', as a signal, to its neighbours. The variable *sending* represents the queue of messages to be sent at time $k\Delta t$. Each message contains the reference trajectory Ref_i^a, deadline M_i, and its ID. This signal is represented by the output world variable *radioOut* which has a certain intensity and is propagated through the environment with a certain diffusion equation not specified here because it is out of the scope of this work. The variable *radioOut* follows the trajectory specified in (6.11): in position p_i it takes the value of the message m with intensity txs, while in all other points of the space where AVs live it takes the value of 0. The value of txs is chosen according to the specifications of the

worldautomaton AV (i,Ref$_i^a$)
world variables LEVEL 0
input avpresent, radioIn,
output radioOut, iampresent,
local variables
internal p_i, Δ_i, k, Δ', deadline, $R_{\text{det,2}}$, RefInt := Ref$_i^a$, rxs, txs
input u, newRef, slow, newVertex
output snapshot, m, sending, $S_{r,i}^k$,
actions
internal *sent, sensor, changeRef*
input *send*
output *receive*
transitions
internal *sensor* (snapshot)
pre $snapshot = avpresent \lceil B(p_i(k\Delta_i), R_{\text{det,2}})$
effect $k = k + 1$
input *send* (m)
effect sending=sending $\cup\{m\}$, deadline = $k + \Delta'$
internal *sent* (m)
pre $m \in$ sending $\wedge$ $k =$ deadline
effect sending=sending $-\{m\}$, deadline = ∞
output *receive* (m)
pre $\exists\ l$ s.t. $(m, l) \in$ radioIn$(p_i(k\Delta_i)), l > rxs\ \wedge m \notin S_{r,i}^k \wedge m$ is for me
effect $S_{r,i}^k$ = $S_{r,i}^k \cup\{m\}$
internal *changeRef*
pre NoDeadline
effect RefInt=Ref$_i^{a+1}$
trajectories

$$\dot{p_i}(t) = u(t); \tag{6.10}$$

$$\text{radioOut}(t, p) = (p = p_i(k\Delta_i)?\{(m, txs)|m \in \text{ sending}\} : \emptyset); \tag{6.11}$$

$$k \leq deadline;\ \text{iampresent}\ (k, p) = (p = p_i(k\Delta_i)?1 : 0); \tag{6.12}$$

if slow $\vee$ newVertex *then* $S_{r,i}^k$ = $S_{r,i}^k$ - m;

Figure 6.5: Autonomous Vehicle world automaton.

communication equipment (for example $txs = R_{\text{det},2}$).

Equation (6.12) describes the trajectory of the boolean output world variable *iampresent*: it takes value 1 where the AV is located (i.e. at position p_i) and value 0 in all other points in the space. This is a restricted version of a more general situation where sensors have more capabilities: for instance, the variable *iampresent* can represent a colour (if the sensor is a camera) with function values in a subset of $\mathbb{Z}^2$, or any other type of sensed characteristic of a vehicle to detect its presence. For the sake of simplicity here we model *iampresent* as a boolean map of presences. Five actions can occur in this WA:

- Action *sensor* represents the scanning of the environment, to detect neighbours each Δ_i time steps, different for each AV. It produces a so-called variable *snapshot*, which is a map of the neighbours' positions: it is the projection of the input world variable *avpresent* (sent by the environment) on a ball $B(p_i(k\Delta_i), R_{\text{det},2})$ centred in $p_i(k\Delta_i)$ and of radius $R_{\text{det},2}$.

- Input action *send* (randomly occurring) produces the message m and adds it to a set of sending messages called *sending*, while setting a deadline Δ' for sending the message (in ideal conditions the messages are instantly sent i.e. $\Delta' = 0$).

- Action *sent* says message m in set *sending* has been transmitted, and removes it from the set.

- Action *receive* adds a message m to set $S^k_{r,i}$ when a new message is caught from the input world variable *radioIn* (sent by the environment). This message is considered to be successfully received when its intensity is higher than a threshold rxs ($= R_{safe}$ in our case).

- Action *changeRef* sets the internal variable RefInt (carrying the data of the initial reference $\text{Ref}\,^a_i$) to a new reference $\text{Ref}\,^{a+1}_n$ given by the environment (supervisor) when the deadline is not met (variable *NoDeadline* sent by the controller).

6.3.2 Environment

The yard is modelled by the environment WA presented in Figure 6.6. The environment takes as input world variables *iampresent* and *radioOut* sent by each of the AVs, while sending to the AVs output world variables *avpresent* and *radioIn*. Variable *radioIn*

diffuses the signal *radioOut* with a certain propagation law if there is an AV in that position and it is set to $\emptyset$ if there is no AV.

worldautomaton Environment
world variables LEVEL 1
    input iampresent: Bool, radioIn
    output avpresent: Bool, radioOut
local variables LEVEL 0
    input NoDeadline
    output newRef
    internal presence: Bool,
trajectories
    radioIn $= \lambda o.$ if no AV $\emptyset$, else propagation(radioOut);
    presence(t, p) = iampresent(t, p);
    avpresent(t, p) = presence(t, p);
    newRef=(NoDeadline ? compute-new-ref : 0);

Figure 6.6: Environment world automaton.

The reader can notice that the world variables that in the AV automaton were of level 0 here are of level 1. Indeed these variables are used by the environment to interact with the world it creates for the AVs, i.e. the inside level of *worlds*, whereas the AVs use these world variables to communicate with the world where they live, i.e. the outside world. Variable *avpresent* is a boolean map describing the presence of AVs in the area, i.e. all the positions where world variables *iampresent* sent by AVs are true, are combined into *avpresent*.

Notice that the supervisor is included into the environment to give a more realistic and natural description of the supervisory action, which basically consists in changing the AVs references when the deadline is not met.

6.3.3 Estimator

The role of the estimator is basically to collect all the neighbors of its corresponding AV which are possibly creating a collision risk. Hence the estimator takes as input the variable *snapshot* sent by the AV and collects all AVs in the *snapshot* in a set $S^k_{d,i}$ (according to the sensor specifications defined in Section 6.1.2). For each $\text{AV}_j \in S^k_{d,i}$ the estimator checks if condition (6.8) is satisfied and if so AV_i includes AV_j in the conflict

worldautomaton Estimator
local variables LEVEL 0
internal $CG_i(k\Delta_i)$
input snapshot, k, Δ_i, $S^k_{r,i}$
output collision: Bool, $S^k_{d,i}, S^k_{c,i}$
actions
internal *collisionrisk*
transitions
internal *collisionrisk*
pre $S^k_{c,i} \neq \emptyset$
effect collision = true
trajectories
$S^k_{d,i} = \{$all AVs in snapshot with ID$\}$;
for each AV$_j \in S^k_{d,i}$
if (AV$_j \in S^k_{r,i} \wedge t_{j,in}(k\Delta_i) > t_{i,ex}(k\Delta_i) \vee t_{i,in}(k\Delta_i) > t_{j,ex}(k\Delta_i)) \vee$
(AV$_j \in S^k_{nr,i} \wedge \tilde{t}_{j,in}(k\Delta_i) > t_{i,ex}(k\Delta_i) \vee t_{i,in}(k\Delta_i) > \tilde{t}_{j,ex}(k\Delta_i)$)
include AV$_j$ in $CG_i(k\Delta_i)$
End if
End for
$S^k_{c,i} = CG_i(k\Delta_i)$

Figure 6.7: Estimator world automaton.

group $CG_i(k\Delta_i)$. In set $S^k_{c,i}$ all the AVs in $S^k_{d,i}$ that are also included in $CG_i(k\Delta_i)$. The estimator raises the flag *collision* if the set $S^k_{c,i}$ is not empty. In Figure 6.7 the automaton of a sensor is shown.

6.3.4 Controller

The automaton of controller CA_i is represented in Figure 6.8. It receives $S^k_{r,i}$ from the AV_i and information, $S^k_{d,i}$ and $S^k_{c,i}$ from the estimator. The controller nominally returns a control velocity (intensity and direction) $u(k\Delta_i)$ following the dynamics described in last line of (6.13).

The controller CA_i checks if any from the detected AVs $\in S^k_{c,i}$, as described in Figure 6.8, is too near and if so it performs a safety manoeuvre (it stops). Otherwise it chooses the most "risky" $AV_j \in S^k_{c,i}$ (the AV with the smallest $t_{j,in}(k\Delta_i)$ - if $AV_j \in S^k_{r,i}$ or $\tilde{t}_{j,in}(k\Delta_i)$ - if $AV_j \notin S^k_{r,i}$) and determines the priority based on the communication attempts results (fail or success). If the AV_i has lower priority it performs a manoeuvre: if communication is established the controller chooses between slowing down or adding a vertex to the trajectory (depending on the introduced delays T_{as} and T_{ar}, respectively); if there is no communication the controller chooses the slowing down manoeuvre. Then it sends the velocity $u(k\Delta_i)$ to the AV automaton. If the original deadline is not met, the controller sends a signal (*NoDeadline*) to the environment that assigns a new trajectory for the corresponding AV.

Notice that each AV in the world uses world variables to communicate its presence to the environment and the environment uses world variables to communicate to all the AVs the locations of the other AVs. Moreover the environment is a means for radio signals to propagate, hence all the AVs catch the signals sent by others in the environment.

6.4 Validation examples

The system described in Section 6.1 is modelled using the world automata framework. Since we developed the model and the rule-based distributed controller above we now validate this design. Thanks to the world automata framework we are able to define the layered architecture of the entire system and, without loss of generality, to have a

worldautomaton CA_i
local variables LEVEL 0
internal priority, slow, newVertex, R_{safe}, stop, restart, v_{max}, $t_{D,new}$, p_j, v_j
input $S^k_{d,i}, S^k_{c,i}, S^k_{r,i}, k, \Delta_i$
output NoDeadline, $u(k\Delta_i)$
actions
internal *slowdown with communication, modify, slowdown without communication*, *deadline, safety, resume*
transitions
internal *slowdown with communication*
pre $AV_j \in S^k_{r,i} \wedge AV_j \in S^k_{c,i} \wedge \neg$priority $\wedge\, T_{as} < T_{ar}$
effect slow-com = true
internal *modify*
pre $AV_j \in S^k_{r,i} \wedge AV_j \in S^k_{c,i} \wedge \neg$priority $\wedge\, T_{ar} < T_{as}$
effect newVertex = true
internal *slowdown without communication*
pre $AV_j \notin S^k_{r,i} \wedge AV_j \in S^k_{c,i} \wedge \neg$priority
effect slow-nocom = true
internal *deadline*
pre $AV_j \in S^k_{c,i} \wedge\ t_{D,new_i} > t_{D_i}$
effect NoDeadline = true
internal *safety*
pre for some $AV_j \in S^k_{c,i}$ distancefrom$(AV_j) \leq R_{safe}$
effect stop = true
internal *resume*
pre $S^k_{d,i} = \emptyset \wedge$ stop
effect restart=true
trajectories
For $AV_j \in S^k_{c,i}$ with smallest time $= (AV_j \in S^k_{r,i}\ ?\ t_{j,in}(k\Delta_i) : \tilde{t}_{j,in}(k\Delta_i))$
priority $= (AV_j \in S^k_{r,i}\ ?$ rule$_{pc}(M_i, M_j) :$ rule$_{pn}(\tilde{p}_j(k\Delta_i), \overrightarrow{\tilde{v}}_j(k\Delta_i), \tilde{t}_{j,in}(k\Delta_i)))$;
End for

$$u(k\Delta_i) = \begin{cases} \text{slowdown-algorithm nr 1} & \text{if slow-com} \\ \text{newvertex-algorithm} & \text{if newVertex} \\ \text{slowdown-algorithm nr 2} & \text{if slow-nocom} \\ 0 & \text{if stop} \\ \text{restart-algorithm} & \text{restart} \\ \text{nominal}(v_{max}) & \text{else} \end{cases} \tag{6.13}$$

$t_{D,new_i} =$ compute-new-deadline-time$(u(k\Delta_i))$

Figure 6.8: Controller world automaton.

close link with the implementation itself. Defining already from the beginning functions and associated input/output parameters made the development of the control algorithm straightforward. Finally, using the concept of local and global variables makes the code cleaner and easier to debug. The model was implemented in Matlab and Java using the Object-Oriented Programming paradigm [65], similarly to the platoon based model example presented in Section 3.3.

6.4.1 Validation results

Since the case without communication is more challenging from the control point of view, all the scenarios have the communication disabled (intensity $txs = 0$). The AVs have the same characteristics: $rxs = R_{safe} = 5$, $R_{det,1} = 10$, $txs = R_{det,2} = 50$, vector speed $V = \{0, 5, 10, 15, 20, 25, 30\}$ [km/h], communication delay $\Delta' = 0$ and with a detection noise uniformly distributed on the open interval $(0, 4)$. Each scenario is run 1000 times and a histogram of the end time T_i^2 for each AV is presented. Our aim is to investigate the probability distributions of the actual end time with respect to the expected end time due to the decision made by each local control. Therefore, the time reserve for each task was set to zero i.e. $T_i^2 = t_{D_i}$. Before presenting the following examples, remember that we use a very pessimistic worst case scenario in which the communication between the AVs is always disabled.

- Figure 6.9 shows a scenario with 3 AVs where $T_1^1 = 0$, $T_2^1 = 6$, and $T_3^1 = 10$ are the start times and $T_1^2 = 49.47$, $T_2^2 = 55.47$, and $T_3^2 = 34$ are the expected end times of the mission. The expected end time is computed as the time that it takes AV_i to perform the task under ideal conditions (AV_i alone in the yard driving at full speed $v_{max} = 30$[km/h]).

  The end time histograms for each AV are presented in Figure 6.10. It can be observed that most of the time the end time T_i^2 encountered by each AV when using the control algorithm is close to the expected end time.

  Due to the noisy detection in the sensors several situations can occur. For example, if AV_1 sees AV_2 the following can happen:

  - AV_1 detects AV_2 and according to the priority rule 1:

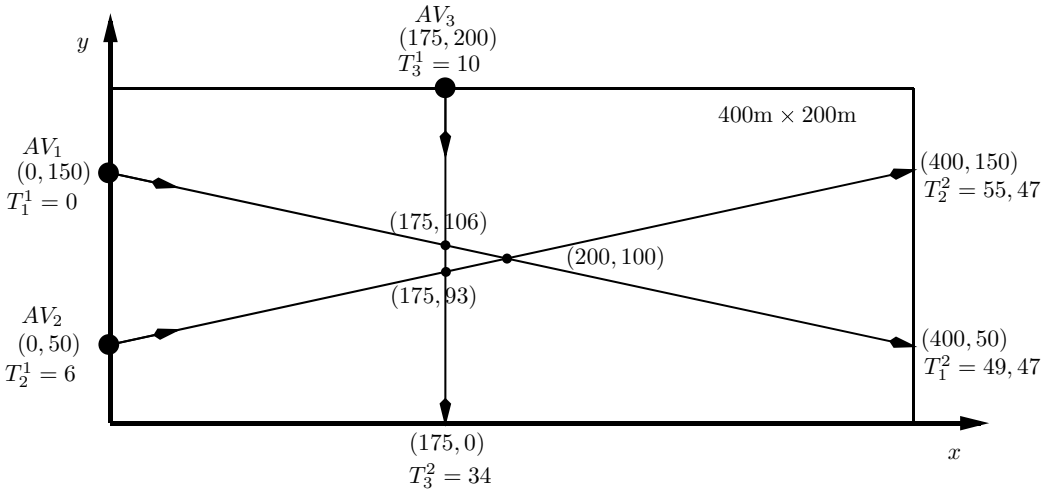


Figure 6.9: Scenario with 3 AVs without communication

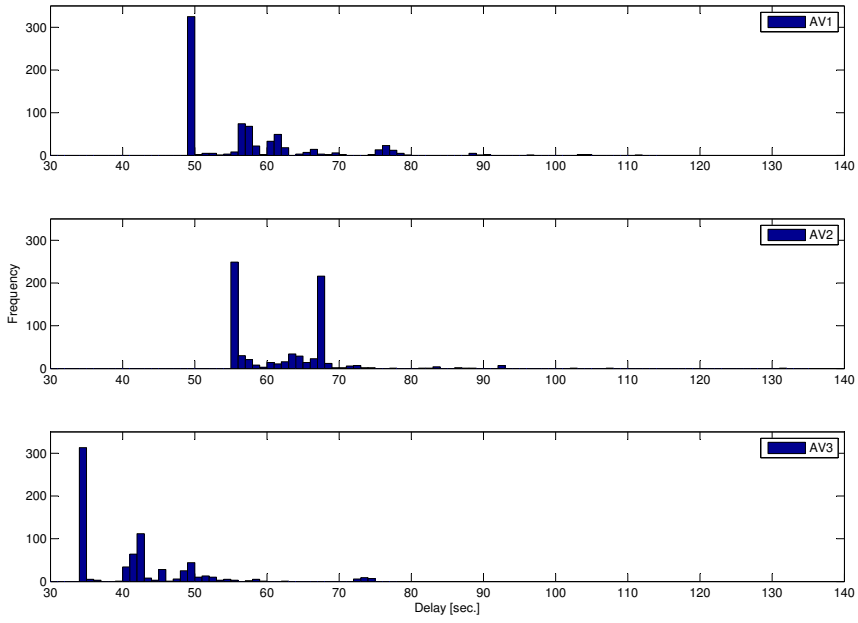


Figure 6.10: The delay histograms of the first example

* AV_1 high and AV_2 low (case.A), or;
* AV_1 low and AV_2 how (case.B), or;
* *undecided* then use priority rule 2: AV_1 low and AV_2 high;

– (case.A) if AV_1 high, goes in front of AV_2, detects AV_3, and according to the priority rule 1:
* AV_1 low and AV_3 high; (case.C)

– (case.B) if AV_2 high, goes in front of AV_1, detects AV_3, and according to the priority rule 1:
* AV_2 low and AV_3 high, or;
* AV_2 high and AV_3 low, or;
* *undecided* then use priority rule 2: AV_2 high and AV_3 low;

– (case.C) if AV_3 high, goes in front of AV_1, detects AV_2 and according to the priority rule 1:
* AV_2 low and AV_3 high, or;
* AV_2 high and AV_3 low, or;
* *undecided* then use priority rule 2: AV_3 low and AV_2 high.

In the case of the first scenario and for the given settings (measurement noise, $R_{\text{det,2}}$, $R_{\text{det,1}}$, end R_{safe}), we encountered 169 deadlocks, as defined by Definition 6.10, out of 1000 runs. By using the permanent communication channel between the supervisor and AVs, the supervisor solves the deadlock.

- The second scenario takes in consideration 4 AVs having collision trajectories as shown in Figure 6.11. The $T_1^1 = 1$, $T_2^1 = 1$, $T_3^1 = 10$, and $T_4^1 = 0$ represent the start times and $T_1^2 = 49$, $T_2^2 = 49$, $T_3^2 = 34$, and $T_4^2 = 48$ are the expected end times of the second scenario. The results are shown in Figure 6.12.

  Similar with the first example, many possible situations can appear also in this case due to the detection noisy in the sensors.

For the second scenario, the number of deadlocks was 0 out of 1000 runs. This fact leads to the conclusion that the supervisor should allocate trajectories along a sort of "virtual" lanes parallel with the quay. Certainly the conclusion must be further validated

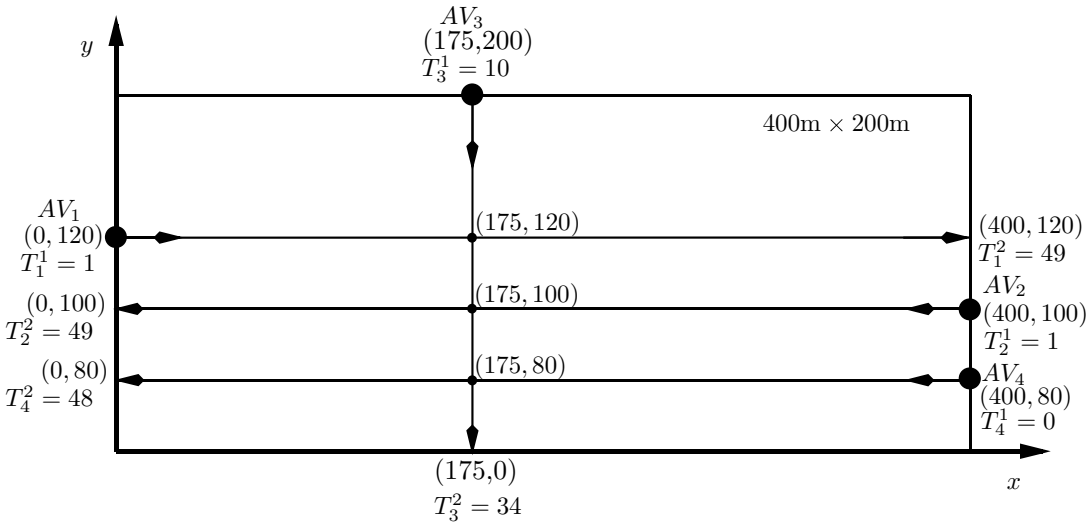


Figure 6.11: Scenario with 4 AVs without communication

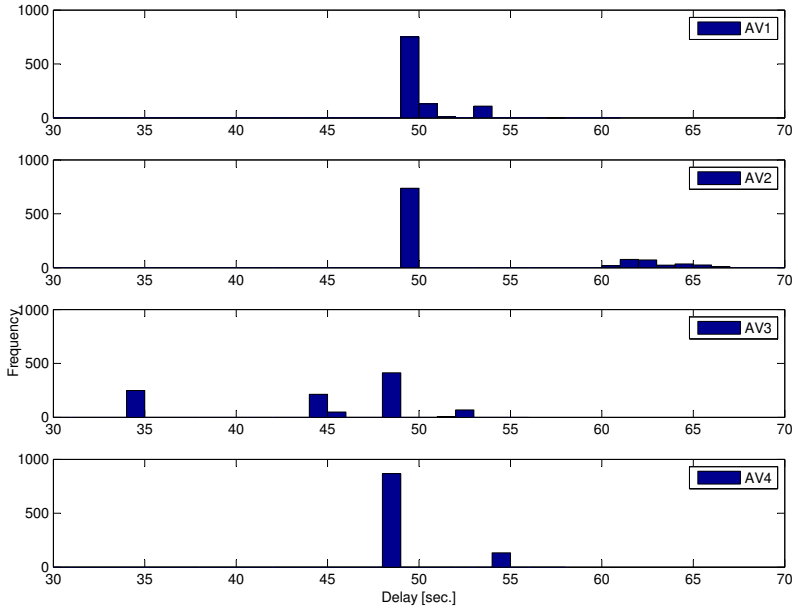


Figure 6.12: The delay histograms of the second example

for more scenarios with different settings. This remark opens the possibility to future research of the trajectory allocation mechanism used by the supervisor.

The interested reader can see different animations of the above presented simulations at https://vimeo.com/colavoidexamples.

6.5 Summary

The modelling and autonomous control problem of autonomous vehicles moving in the free space area of a container terminal is presented in the current chapter. The coordination of automated vehicles is achieved by locally applying a set of priority rules, which can be seen as determining the leader in case of conflict. The communication and computational complexity are thereby significantly reduced compared to a centralised supervisory control solution. We start by giving a description of the system together with the proposed control architecture. The control problem is detailed for two possible cases, with and without communication, followed by the possible manoeuvres that can be made by the AVs to avoid a possible collision. The model of the system is introduced using the world automata framework. The description of the simulation environment and the obtained results end the chapter.

Chapter 7

Conclusions and future work

The results of the previous chapters are summarised in the current section. We give several observations about the obtained results, and possible research directions for future extensions of the work.

7.1 Conclusions and observations

The thesis investigates the modelling, distributed estimation and control of interacting hybrid systems. In the case of distributed control of such systems, an approach where local controllers (also called agents) communicate with their neighbours is proposed and leader-follower - type algorithms that achieve coordination already at this local or low in hierarchy level are devised. We have studied two such interacting hybrid systems, namely, urban traffic networks and control of autonomous vehicles in a container terminal. The complexity and the stochastic behaviour of the examples make the underlying theoretical problem very difficult. Last, but foremost, the goal was to learn from this kind of examples and aim for a "general theory" for the estimation and control of networks of interacting hybrid systems.

The keystone for any advanced control application is the model as a representation of the real system. Therefore, with the focus on DEDS, we revised different modelling frameworks in Chapter 2. Some of the introduced frameworks are later used to model and simulate the two examples. We found that such models of DEDS are more suitable to represent complex systems that are encountered in practice than closed-form models (representations that can be expressed analytically in terms of a finite number of

certain ”well-known” functions). Models can vary from simple forms up to complex representations that reach the desired accuracy.

In Chapter 3, we have developed a novel model using platoons as an abstraction for modelling urban traffic networks. After a classification of existing types of urban traffic models, we showed that vehicles typically travel closely together at approximately the same speed, behaving as one coherent platoon, due to overtaking limitations, and as a result of the interaction with the red/green cycle of traffic lights. An analysis was performed based on raw real traffic measurements taken over 40 days in the area of Dendermonde, Belgium. Results showed that aggregating urban traffic flow in entities called platoons enables simulation tools faster than the microscopic models representing individual vehicles, while explicitly representing the heterogeneity characterizing urban traffic, which is not possible with a macroscopic model. Matlab was used for prototyping the platoon based model.

Even though the ”discretisation” of urban traffic in platoons is conceptually attractive and easy to understand, the model conceptualisation was not a straight forward process. The computational load of our model is strictly determined by the number of events, and thus larger platoons (which form when the load increases) do not change anything. The estimation and control applications developed later in this thesis are feasible thanks to the efficient platoon based simulations.

In Chapter 4, we have developed estimators for urban traffic networks using the platoon based model. We proposed two particle filter algorithms for the estimation of the queue sizes at intersections and the platoon locations along the links of the network. The particle filter is an exceedingly general dynamic and measurement based estimation method, that easily accommodates sensor failures, data coming from different sources, and any type of nonlinear/hybrid system model. After developing a centralised particle filter algorithm, presented in Algorithm 4.1, we took advantage of the modular structure of our model to propose a distributed particle filter scalable to very large networks. Algorithm 4.2 lists the distributed particle filter algorithm. We validated our estimation algorithms using synthetic data produced by a third party microsimulator SUMO [3]. This verifies that the method is robust against some model uncertainties, which include the many differences between SUMO and our reduced model as well as rare events like accidents. In the particle filter framework, it is difficult to know the most efficient number of particles which leads to the best “accuracy against efficiency“ trade-off. More

difficult is to understand where you find yourself in this trade-off situation whereas this is not the case for a Kalman filter.

In general, a particle filter framework has a high computational burden given by the number of particles (possible sampled system trajectories weighted by the likelihood given to the observed sensor outputs) and the used model. For larger networks, the computational complexity per particle grows (e.g. linearly with the number of platoons in the network), and the number of particles must also be increased to ensure a good coverage of the increased number of possibilities (difficult to quantify). We have shown however that the particle filter can be efficiently parallelised by partitioning the network into subnetworks. The distributed estimation solution proposed in Chapter 4 enables the use of distributed feedback control strategies, that allow the traffic lights react to the networks state as estimated by the local particle filters. For robustness reasons, such control strategies can for instance guarantee good performance of all the particles whose weights are above some threshold.

In Chapter 5, we have proposed a new distributed feedback control framework for the control of urban traffic networks under intermediate load for two-way traffic. Using a model based feedback framework and local state information (platoons approaching the intersection), local controllers (agents) select the switching times of traffic lights of the network. Although in practice the state information would come from a distributed particle filter estimator like the one presented by us in Chapter 4, for independent evaluation we have currently programmed the controller on the basis of perfect measurements of platoons at intersections. For simplicity, in our work, left turning traffic in the intersections is disabled but this does not represent a limitation of our approach. The aim is to simultaneously minimise the waiting time of all vehicles by combining a real-time reaction to traffic arriving at local intersections with a green wave type objective, where platoons of vehicles would travel through consecutive intersections without having to stop. Using the leader-follower paradigm, the coordination between agents of a network is created. A supervisory layer selects from the agents of the network using the ”heaviest traffic load” criterion the leader agents and theirs followers. Formally implementing and testing the supervisory layer is left for future work. The follower agents take into account the implications of their switching choices on the timing of traffic sent to neighbouring leader agents.

We started our presentation with a comparison between our method and different

state-of-the-art adaptive control solutions that are currently in wide use and with some theoretical or fundamental similarity to our work. Thus, in Section 5.1 we described similarities and differences with OPAC-[31], SCOOT-[37], SCATS-[49], RHODES-[61] and different papers where platoons of vehicles are proposed to be used in the control of traffic lights. The presentation continued with a preliminary control analysis (using a fluid flow representation) for different traffic intensities (saturated, intermediate, and very light) at a single intersection. A practical lesson, later used as a heuristic when predicting future switching behaviour in our framework, was that switching faster in open loop is usually better than any other strategies. As a consequence of the latter remark and to cope with modelling limits, we imposed a minimal green time between consecutive phases. For intermediate traffic conditions, there are significant random (but predictable according to the platoon based model) gaps between platoons. Flexible, control actions for this type of traffic are advisable and therefore a good approach would be to select actual switching times instead of using green cycle and split. Therefore, we proposed using only switching times as control actions for intermediate traffic conditions.

In Section 5.3.1, we defined an optimisation problem (5.5) that, if solved by each agent over a given time horizon, leads to deciding whether to extend or not the current green phase based only on local information. Insights were given about calculating the total optimisation cost (5.4) and on some practical details about solving the optimisation. A simple example (three constant average rate inflows and one precise platoon inflow) was also presented where the cost is not necessarily a convex function depending on the timing of arriving platoons. The local optimisation problem was implemented in Algorithm 5.1. Since our final aim was to introduce coordination between neighbouring agents based on the leader-follower paradigm, we proposed a not-following cost in Section 5.3.2. The cost implemented in Algorithm 5.3 shows the penalisation paid by follower agents for deviating from the desired schedule sent by the leader. The proposed penalisation was computed based on how the leader queues would evolve over one nominal cycle, for average traffic and with a starting queue zero at leader and follower (justified for an outflow higher than twice the inflow value). Algorithm 5.4 lists the code of the leader-follower framework and calls Algorithm 5.1 for both, leaders and followers, and Algorithm 5.3 only if the agent is a follower.

We made different experiments to verify and validate our framework using a network

with five intersections interconnected in a star topology. We aimed to show that by enhancing the coordination between agents better performance can be obtained instead of a fully distributed solution (not-following cost zero). Compared with a nominal green wave based switching scenario, our approach creates a dynamic green wave that can adapt to traffic variations. In Figure 5.24, it seams that with $\omega = 1000$ (closer to the nominal green wave - followers pay a very high cost for not following and the local optimisation is almost dropped) the set-up performs worse compared to the cases when using $\omega = 100$ or $\omega = 500$. Our method needs a supervisory layer used to select the leader agents based on a predefined criterion (e.g. "heaviest traffic load" criterion in our case). Despite this fact, the leader-follower paradigm limits the communication (from leaders to followers and not vice-versa) while still preserving the flexibility in making local decision.

In Chapter 6, we presented the second case study on modelling and control for collision avoidance of autonomous vehicles used in a container terminal (free area between the quay and the rows of containers). What made the problem challenging were the asynchronous interactions between the AVs, the uncertainty present in the detection and in the communication. Firstly, we developed a clear representation of our proposed architecture. In this way, more details can be easily introduced specifically to each level (environment, AVs, control level, and nevertheless the supervisor). Secondly, we proposed a local rule-based controller used for avoiding collision with other AVs. The control algorithm, described in Section 6.2, is able to cope with asynchronous interactions, the uncertain information coming from the sensors of the AV and with an unreliable communication channel between AVs. The output of the controller of each AV determines the priority of that AV regarding the other(s) detected in its range (highest priority vehicle AV_{high} as a leader and AV_{low} as a follower). We presented the model of the case study in Section 6.3.

Using the model with the controller, we determined the empirical distributions of the expected delays for the AVs. These can be used to construct theoretical probability distributions for a queueing analysis. Performing such analysis is useful in practical situations, e.g. in the case of a container terminal when, many AVs have the same destination and the required order in which the AVs arrive cannot be subject to changes. The same destination for many AVs can be the quay cranes loading or unloading a berthed vessel. We tested our approach with different scenarios and the results are

presented in Section 6.4. The analysis can be easily performed and extended using the presented model.

In conclusion, we developed two control approaches for the coordination of urban traffic networks and automated guided vehicles. For the first approach, we were able to obtain coordination between the agents of the network and thus to control the entire network by selecting some leader agents that require follower agents from the neighbouring intersections to coordinate their actions with given leader decisions. As a second approach, the coordination of automated guided vehicles was achieved by locally applying a set of priority rules, which can be seen as determining the leader in case of conflict. Both approaches minimise the necessity of supervising the network at a centralised level. This can be crucially important for scalability to very large networks of any kind.

In general, for network control problems (e.g. logistics networks, manufacturing networks, irrigation channels, etc.), the leader-follower principle can be used between the local-decision makers to obtain coordination. We can extend the leader-follower principle at the different layers (e.g. between controllers of different regions that contain local-decision makers) combined with a global supervisor-rule at the highest layer. By applying this extension, a complicated network can be hierarchically controlled and the design of the supervisors from each layer (i.e. assuming that more than two layers are necessary) can be simplified. Such scenario could be used for example in controlling the manufacturing machines throughout a factory (the lowest layer represents local agents controlling individual machines, the next layer of agents controlling the machines located in same department, agents controlling multiple departments from the same building, and so forth). Of course, selecting the penalty cost of the followers found at any level remains to be solved and is specific for each problem.

We also developed solutions for the estimation of urban traffic networks. Starting from a centralised solution, we have shown that a distributed particle filter can in some situations, like the present one, allow to make savings in exchanged data. We observe that particle filters pick up a trend correctly, but there is still space for improvements. The algorithms do not seem to give good conclusions when the uncertainties (e.g. the hybrid uncertainties in motion dynamics and in sensors in our case) are more predominant in the generation of particles and less in evaluating their likelihood (i.e in our case we only included the sensor uncertainties).

7.2 Future work

In this section, we list a few open problems left for future extensions of our work. The solutions proposed in this thesis were focused on two case studies (i.e. urban traffic networks and the control of autonomous vehicles in a container terminal), but can be used for a larger class of systems. For each chapter of the thesis, we give possible research directions that could be followed.

7.2.1 Modelling of urban traffic networks

The idea behind the platoon based model could also be used for other systems e.g. to model the arrival of units in batches at the machines from a manufacturing networks. Future research directions include:

- the analysis of a more complete set of measurements to study the effects of platoons as an abstraction; an open problem is to prove the existence of a fundamental diagram for urban traffic using platoons - relevant work is presented in [21] and [33];
- we propose further abstractions (e.g. the classification according to the flow intensities of urban traffic) as closer representations to the fluid flow model and that could produce even faster simulations than our model. The improved computational efficiency of such representations, implemented as a discrete event simulator, comes from the further reduction of the number of events that have to be executed, compared with the implementation of our model.

7.2.2 Distributed estimation of urban traffic networks

So far, our solution was to parallelise the computational burden over the $\mathcal{N}_R$ local estimators of the network, aiming to obtain approximately the same quality of a centralised approach. Some global coordination is still necessary for resampling, but the most heavy variables - local measurements and states - never need to be broadcast. Furthermore, these global operations need not necessarily take place at every iteration, but only when the output is needed. Future research directions are:

- optimizing parameters like the number of particles and the frequency of resampling;
- a validation with larger sets of real data to add/remove more relevant aspects in the modular description of the platoon based model;
- introduction of new data sources (e.g. video cameras, radars) and investigation of the estimates;
- adding more flexibility by adapting the local number of particles to actual requirements of each local estimator (e.g. using $M \cdot \mathcal{N}_\mathcal{P}$ particles in a busy region to represent $M > 1$ possible local situations compatible with a single particle group i of the other regions);
- another distributed resampling approach like the one proposed in [72] where each local estimator sends to its neighbours only the i most heavy variables (with the highest weights) out of $\mathcal{N}_\mathcal{P}$ and the resampling is performed on $\mathcal{N}_\mathcal{P} + i \cdot$#number of neighbours. The global coordination for resampling is avoided for this approach, which makes it more robust against network failures than in our case, where a central node is necessary;
- presenting platoon location estimates besides queue sizes, as we already mentioned in Section 4.4.4. For example, one particle that has five vehicles in a queue and another particle which has only two vehicles in a queue, but three other vehicles just microseconds away from joining/leaving that queue, represent in practice the same traffic state. In fact, this variation is very likely to appear since in reality vehicle speed is known to vary, while our model, when grouping vehicles into a platoon, makes the approximation that all the vehicles have the same speed. Therefore, a better way to evaluate the likelihood of particles would be to compare the deviations in time of expected vehicle detections (according to the particle) and actual sensor signals. Firstly, the visualisation of the platoon location estimates is not straightforward to be made in a way that is useful for control applications. More importantly, in the presence of such hybrid uncertainties in motion dynamics and in sensors, a more advanced procedure should be investigated for sharing the uncertainty between generating particles and evaluating their likelihood: the weight should not just include sensor uncertainties, but

also account for some of the modelling uncertainty in order to lower the burden on the particle sampling.

7.2.3 Distributed control of urban traffic networks

The developed solution is based on feedback control agents that coordinate their actions for the control of urban traffic networks under intermediate load for two-way traffic. The coordination is done by selecting some leader agents that require neighbouring agents (called followers) to coordinate their actions with given leader decisions. The selection of the leader agents is done by a supervisory layer based on a criterion (e.g. "heaviest traffic load" criterion) but the development of this layer is not covered by the current work. In the interest of independent evaluation, we have used the controller on the basis of perfect measurements of platoons at intersections. Future research directions include:

- increasing the number of switching points that are explored before assuming a nominal future. The complexity of the optimization (maximal number of scenarios to be enumerated) increases like $(\overline{(G_j - \underline{G_j})}/\Delta_s)^M$ for M switching actions to be optimized. As such, this can quickly become too complex for real-time operation, but solutions can also be found for $M > 1$. A longer Δ_s ($> 1[sec.]$) reduces the number of scenarios. After the first switching point, the granularity of the possible next switching points can be decreased, which results in a further reduction of the number of scenarios.

- using the output of the distributed estimator developed in Chapter 4 in combination with the distributed controller presented in Chapter 5. The question that needs to be answered is: which particles of the local particle filter to use in the control design?

- running more simulations (currently we have investigated with 20 or 30 simulations for each ω) using other traffic inflow scenarios;

- test different network topologies to better understand the influence of the leader-follower concept over the performance of the entire network;

- include new penalties in the local cost formula (e.g. penalisation for the remaining queue at the end of the green);

- further development of the supervisor (e.g. based on an abstracted model as suggested in Section 7.2.1) that allocates the leaders and their followers in a network and the best practice of choosing the leader follower configuration in a real urban traffic network; this demands to run the controller in a real environment and to investigate different strategies to select the leaders for a given network;

- another approach that can be used, where an upper layer is not necessary, is the cooperative distributed control. In this approach, the local agents still have to exchange local messages until a consensus is reached (i.e. in practice after a few communication cycles). We will usually find a sub-optimal solution as a consequence of exchanging only local messages instead of some global network broadcast. The most important operational difference with our approach is that there is no preassigned ”leader” anymore. Thus, which agent has more influence on the control policy should be a result of the cooperative consensus strategy;

7.2.4 Modelling and control for collision avoidance of autonomous vehicles

We also investigated the coordination of automated guided vehicles used in a container terminal. Each vehicle is controlled by an agent which must ensure that a delivery task is completed on time, while at the same time guaranteeing safe operation by automatically avoiding collisions. We set up a rule-based controller governing the movement of each local agent, such that a set of agents observing each other in a situation of potential conflict apply priority rules that lead to their coordination. A supervisory layer is used to assign tasks and to resolve potential conflicts that could not be solved by the agents. The development of this layer is not covered by the current work. Some open problems left for further investigation are:

- the development of the supervisory layer that assigns tasks, to further include conflict resolution in case of failure of the local rule-based controllers;

- a comparison with other approaches;

- due to the unlimited number of possible scenarios, a tool could be developed

to generate classes of scenarios that are not feasible for a given set of input parameters;

- the development of tools that together with our model could automatically generate scenarios for which the set of rules used by the controller are not feasible.

7.3 Summary of the original contributions of the thesis

The results of the research were disseminated in the following journals and conference papers:

2013	• *Nicolae Emanuel Marinică*, Marta Capiluppi, Roberto Segala and René Boel, **"Distributed collision avoidance for autonomous vehicles: modeling and control"**, (ongoing work), 2013. • *Nicolae Emanuel Marinică*, Sarlette Alain and René Boel, **"Distributed Particle Filter for Urban Traffic Networks using a Platoon Based Model"**, IEEE Transactions on Intelligent Transportation Systems, vol.14, no.4, p.1918-1929, 2013.
2012	• *Nicolae Emanuel Marinică* and René Boel, **"A leader/follower approach for distributed coordination of interacting components"**, Proceedings of the 20$^{\text{th}}$ International Symposium on Mathematical Theory of Networks and Systems (MTNS), 2012. • *Nicolae Emanuel Marinică* and René Boel, **"Platoon based model for urban traffic control"**, Proceedings of the American Control Conference (ACC), p.6563-6568, 2012. • *Nicolae Emanuel Marinică*, Marta Capiluppi, Jonathan Rogge, Roberto Segala and René Boel, **"Distributed collision avoidance for autonomous vehicles: world automata representation"**, Proceedings of the 4$^{\text{th}}$ IFAC Conference on Analysis and Design of Hybrid Systems (ADHS), p.216-221, 2012.
2011	• *Nicolae Emanuel Marinică* and René Boel, **"Particle filter state estimator for large urban networks"**, Proceedings of the 2011 Australian Control Conference (AUCC), p.374-380, 2011.

	• *Nicolae Emanuel Marinică* and René Boel, **"CSP for coordination of distributed systems"**, Book of abstracts of 30th Benelux meeting on Systems and Control, Lommel, Belgium, 2011.
	• *Nicolae Emanuel Marinică* and René Boel, **"Particle filter for platoon based models of urban traffic"**, Proceedings of the 15th WSEAS international conference on systems, Corfu Island, Greece, p.42-47, 2011.
2009	• *Nicolae Emanuel Marinică* and René Boel, **"Particle filter state estimator for large urban networks"**, Book of abstracts of 28th Benelux meeting on Systems and Control, Spa, Belgium, 2009.

and various project or national meetings:

2011	• *Nicolae Emanuel Marinică* and René Boel, **"Hierarchical feedback coordination of urban traffic networks"**, CON4COORD review meeting, Porto, Portugal, 2011.
	• *Nicolae Emanuel Marinică* and René Boel, **"Using the platoon based model for distributed estimation and control"**, CON4COORD 9th project meeting, Verona, Italy, 2011.
2010	• *Nicolae Emanuel Marinică* and René Boel, **"Updates on the platoon based model"**, CON4COORD 8th project meeting, Ghent, Belgium, 2010.
	• *Nicolae Emanuel Marinică* and René Boel, **"Updates on particle filter for state estimation in urban traffic"**, CON4COORD 7th project meeting, Delft, The Netherlands, 2010.
	• *Nicolae Emanuel Marinică* and René Boel, **"Particle filter for state estimation in urban traffic."**, CON4COORD 6th project meeting, Volos, Greece, 2010.
2009	• *Nicolae Emanuel Marinică*, Jonathan Rogge and René Boel, **"Platoon based models for distributed control of urban traffic networks"**, CON4COORD 5th project meeting, Verona, Italy, 2009.
	• *Nicolae Emanuel Marinică*, Jonathan Rogge and René Boel, **"Antwerp Container Terminal Automation"**, CON4COORD 4th project meeting, Antwerp, Belgium, 2009.

• *Nicolae Emanuel Marinică*, Jonathan Rogge and René Boel, **"Antwerp Container Terminal Study"**, CON4COORD 3rd project meeting, Cyprus, 2009.

• *Nicolae Emanuel Marinică* and René Boel, **"A platoon based model for urban traffic networks: identification, modeling and distributed control"**, poster presentation at Interuniversity Attraction Pole IAP VI/4 DYSCO Study Day,Leuven, Belgium, 2009.

References

[1] Alur, R. and Dill, D. L.: 1994, A Theory of Timed Automata, *Theoretical Computer Science* **126**(2), 183 – 235. 17

[2] Artimy, M. M., Robertson, W. and Phillips, W. J.: 2004, Connectivity in Inter-vehicle Ad Hoc Networks, *Proceedings Canadian Conference on Electrical and Computer Engineering*, Vol. 1, pp. 293 – 298. 32

[3] Behrisch, M., Bieker, L., Erdmann, J. and Krajzewicz, D.: 2011, Sumo - simulation of urban mobility: an overview, *Proceedings* 3^{rd} *International Conference on Advances in System Simulation*, pp. 55 – 60. 5, 7, 32, 66, 159

[4] Boel, R. and Mihaylova, L.: 2006, A compositional stochastic model for real-time freeway traffic simulation, *Transportation Research B* **40**, 319 – 334. 55

[5] Bolic, M., Djuric, P. M. and Hong, S.: 2004, Resampling algorithms for particle filters: a computational complexity perspective, *EURASIP Journal on Applied Signal Processing* **15**, 2267 – 2277. 58

[6] Braun, R. and Weichenmeier, F.: 2005, Automatic offline-optimization of coordinated traffic signal control in urban networks using genetic algorithms, *Proceedings of the* 12^{th} *World Congress on Intelligent Transport Systems*. 32

[7] Cafieri, S., Brisset, P. and Durand, N.: 2010, A mixed-integer optimization model for Air Traffic Deconfliction, *Proceedings of TOGO 2010*, pp. 27 – 30. 132

[8] Capiluppi, M. and Segala, R.: 2012, Modelling implicit communication in multi-agent systems with hybrid input/output automata, *Third International Symposium*

on Games, Automata, Logics and Formal Verification (GandALF 2012), Electronic Proceedings in Theoretical Computer Science (EPTCS). 24

[9] Capiluppi, M. and Segala, R.: 2013, World automata: a compositional approach to model implicit communication in hierarchical hybrid systems, *Third Workshop in Hybrid Autonomous Systems 2013 (HAS 2013)*, Electronic Proceedings in Theoretical Computer Science (EPTCS). 24, 131

[10] Cassandras, C. G. and Lafortune, S.: 2006, *Introduction to Discrete Event Systems*, Springer-Verlag New York, Inc., Secaucus, NJ, USA. 10, 13, 17, 20, 21, 25

[11] Ceylan, H.: 2006, Developing combined genetic algorithm hill-climbing optimization method for area traffic control, *Journal of Transportation Engineering* **132**(8), 663 – 671. 32

[12] Chan, E., Gilhead, P., Jelinek, P., Krejci, P. and Robinson, T.: 2012, Cooperative Control of SARTRE Automated Platoon Vehicles, *Proceedings of the* 19^{th} *ITS World Congress*. 5, 34

[13] Choffnes, D. R. and Bustamante, F. E.: 2005, Straw - An Integrated Mobility and Traffic Model for VANETs, *Proceedings of the* 2^{nd} *ACM international workshop on Vehicular ad hoc networks*, pp. 69 – 78. 32

[14] Colombaroni, C., Fusco, G. and Gemma, A.: 2009, Optimization of traffic signals on urban arteries through a platoon-based simulation model, *Proceedings of the* 11^{th} *WSEAS international conference on Automatic control, modelling and simulation*, ACMOS'09, pp. 450 – 455. 77, 84

[15] Cowan, R.: 1980, Further results on single-lane traffic flow, *Journal of Applied Probability* **17**(2), 523 – 531.
URL: *http://www.jstor.org/stable/3213041* 34

[16] *Cube Avenue*: n.d.
URL: *http://www.citilabs.com/products/cube/cube-avenue* 32

[17] *Cube Voyager*: n.d.
URL: *http://www.citilabs.com/products/cube/cube-voyager* 32

[18] Daganzo, C. F.: 1994, The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory, *Transportation Research Part B: Methodological* **28**(4), 269 – 287. 32

[19] Daganzo, C. F.: 1995, The cell transmission model, part II: Network traffic, *Transportation Research Part B: Methodological* **29**(2), 79 – 93. 32

[20] Daganzo, C. F.: 1997, *Fundamentals of Transportation and Traffic Operations*, Pergamon-Elsevier, Oxford, U.K. 32, 77

[21] Daganzo, C. F. and Geroliminis, N.: 2008, An analytical approximation for the macroscopic fundamental diagram of urban traffic, *Transportation Research Part B: Methodological* **42**(9), 771 – 781. 29, 30, 164

[22] De Schutter, B. and De Moor, B.: 1998, Optimal traffic light control for a single intersection, *European Journal of Control* **4**(3), 260 – 276. 83

[23] Defoort, M., Doniec, A. and Bouraqadi, N.: 2011, *Informatics in Control Automation and Robotics*, Vol. 85 of *Lecture Notes in Electrical Engineering*, Springer, chapter "Decentralized Robust Collision Avoidance Based on Receding Horizon Planning and Potential Field for Multi-Robots Systems", pp. 201 – 215. 132

[24] Dotoli, M., Fanti, M. P. and Meloni, C.: 2006, A signal timing plan formulation for urban traffic control, *Control Engineering Practice* **14**(11), 1297 – 1311. 32

[25] Douc, R. and Cappe, O.: 2005, Comparison of resampling schemes for particle filtering, *Proceedings of the* 4^{th} *International Symposium on Image and Signal Processing and Analysis*, pp. 64 – 69. 58

[26] Doucet, A., De Freitas, N. and Gordon, N. J.: 2001, *Sequential Monte Carlo Methods in Practice*, Springer. 56

[27] Durrant-Whyte, H. F., Pagac, D., Rogers, B., Stevens, M. and Nelmes, G.: 2002, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, *IEEE Transactions on Signal Processing* **50**, 174 – 188. 57

[28] Durrant-Whyte, H. F., Pagac, D., Rogers, B., Stevens, M. and Nelmes, G.: 2007, An autonomous straddle carrier for movement of shipping containers, *IEEE Robotics and Automation Magazine* **14**, 14 – 23. 132

[29] *Dynameq*: n.d.
URL: *http://www.inro.ca/en/products/dynameq/index.php* 32

[30] *Emme*: n.d.
URL: *http://www.inro.ca/en/products/emme/index.php* 32

[31] Gartner, N. H.: 1983, OPAC: A demand-responsive strategy for traffic signal control, *Transportation Research Record* **906**, 75 – 81. 3, 78, 80, 161

[32] Gazis, D. C. and Potts, R. B.: 1965, The Over-Saturated Intersection, *Proceedings of the Second International Symposium on the Theory of Road Traffic Flow*, pp. 221 – 237. 32

[33] Geroliminis, N. and Daganzo, C. F.: 2008, Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings, *Transportation Research Part B: Methodological* **42**(9), 759 – 770. 29, 30, 164

[34] Geroliminis, N. and Skabardonis, A.: 2011, Identification and Analysis of Queue Spillovers in City Street Networks, *IEEE Transactions on Intelligent Transportation Systems* **12**(4), 1107 – 1115. 30

[35] Gordon, N. J., Salmond, D. J. and Smith, A. F. M.: 1993, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *Radar and Signal Processing, IEE Proceedings F* **140**(2), 107 – 113. 56

[36] Haddad, J., De Schutter, B., Mahalel, D., Ioslovich, I. and Gutman, P.-O.: 2010, Optimal steady-state control for isolated traffic intersections, *IEEE Transactions on Automatic Control* **55**(11), 2612–2617. 84

[37] Hansen, B. G., Martin, P. T. and Perrin, H. J.: 2000, SCOOT Real-Time Adaptive Control in a CORSIM Simulation Environment, *Transportation Research Record: Journal of the Transportation Research Board* **1727**, 27 – 30. 3, 77, 80, 161

[38] He, Q., Larry Head, K. and Ding, J.: 2012, PAMSCOD: Platoon-based arterial multi-modal signal control with online data, *Transportation Research Part C: Emerging Technologies* **20**(1), 164 – 184. 77, 85

[39] Henzinger, T. A.: 1996, The Theory of Hybrid Automata, *IEEE Computer Society Press*, pp. 278 – 292. 21

[40] Hoogendoorn, S. P. and Bovy, P. H. L.: 2001, State-of-the-art of vehicular traffic flow modelling, *Proceedings of the Institution of Mechanical ENgineers, Part I: Journal of Sysmtes and Control Engineering*, pp. 283 – 303. 86

[41] Hopcroft, J. E., Motwani, R. and Ullman, J. D.: 2006, *Introduction to Automata Theory, Languages, and Computation* (3^{rd} *Edition)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 13

[42] Jiang, Y., Li, S. and Shamo, D. E.: 2006, A platoon-based traffic signal timing algorithm for major/minor intersection types, *Transportation Research Part B: Methodological* **40**(7), 543 – 562. 77, 84

[43] Kaynar, D. K., Lynch, N., Segala, R. and Vaandrager, F.: 2006, *The Theory of Timed I/O Automata (Synthesis Lectures in Computer Science)*, Morgan & Claypool Publishers. 19

[44] Krauss, S.: 1998, *Microscopic modeling of traffic flow: investigation of collision free vehicle dynamics*, PhD thesis, DLR Köln, Hauptabteilung Mobilität und Systemtechnik. 66

[45] Kumar, R. and Garg, V. K.: 1995, *Modeling and Control of Logical Discrete Event Systems*, Kluwer Academic Publishers, Boston, USA. 10

[46] Lim, J. H., Hwang, S. H., Suh, I. H. and Bien, Z.: 1981, Hierarchical optimal control of oversaturated urban traffic networks, *International Journal of Control* **33**(4), 727 – 737. 83

[47] Lin, S., De Schutter, B., Xi, Y. and Hellendoorn, H.: 2012, Efficient network-wide model-based predictive control for urban traffic networks, *Transportation Research Part C* **24**, 122–140. 84

[48] Lin, S., De Schutter, B., Xi, Y. and Hellendoorn, H.: 2013, Integrated urban traffic control for the reduction of travel delays and emissions, *IEEE Transactions on Intelligent Transportation Systems* **14**(4), 1609–1619. 84

[49] Lowrie, P. R.: 1992. Sydney Co-ordinated Adaptive Traffic System. NSW Australia: Roads and Traffic Authority. 3, 78, 80, 81, 161

[50] Lynch, N. A. and Tuttle, M. R.: 1989, An Introduction to Input/Output Automata, *CWI Quarterly* **2**, 219 – 246. 19, 27

[51] Lynch, N., Segala, R. and Vaandrager, F.: 2003, Hybrid I/O automata, *Information and Computation* **185**(1), 105 – 157. 23, 24

[52] Marinică, E. N., Capiluppi, M., Rogge, J. A., Segala, R. and Boel, R. K.: 2012, Distributed collision avoidance for autonomous vehicles: world automata representation, 4^{th} *IFAC Conference on Analysis and Design of Hybrid Systems (ADHS)*, pp. 216 – 221. 131

[53] Marinică, N. and Boel, R.: 2011, Particle filters state estimator for large urban networks, *Proceedings Australian Control Conference*, pp. 374 – 380. 55

[54] Marinică, N. and Boel, R.: 2012a, A leader/follower approach for distributed coordination of interacting components, *Proceedings of* 20^{th} *International Symposium on Mathematical Theory of Networks and Systems*. 47, 52

[55] Marinică, N. and Boel, R.: 2012b, Platoon based model for urban traffic control, *Proceedings of American Control Conference*, pp. 6563 – 6568. 39

[56] Marinică, N. E., Sarlette, A. and Boel, R. K.: 2013, Distributed Particle Filter for Urban Traffic Networks Using a Platoon-Based Model, *IEEE Transactions on Intelligent Transportation Systems* **14**(4), 1918 – 1929. 55

[57] Michael, J. B., Godbole, D. N., Lygeros, J. and Sengupta, R.: 1998, Capacity Analysis of Traffic Flow Over a Single-Lane Automated Highway System*, *ITS Journal - Intelligent Transportation Systems Journal* **4**(1–2), 49–80. 5

[58] Mihaylova, L. and Boel, R.: 2004, A particle filter for freeway traffic estimation, *Proceedings* 43^{rd} *IEEE Conference on Decision and Control*, pp. 2106 – 2111. 55

[59] Mihaylova, L., Boel, R. and Hegyi, A.: 2006, Freeway traffic estimation within particle filtering framework, *Automatica* **43**, 290 – 300. 55

[60] Miller, A. J.: 1963, A computer control system for traffic networks, *Proceedings of the* 2^{nd} *International Symposium on the Theory of Traffic Flow, London.*, pp. 200 – 220. 103

[61] Mirchandania, P. and Headb, L.: 2001, A real-time traffic signal control system: architecture, algorithms, and analysis, *Transportation Research Part C: Emerging Technologies* **9**(6), 415 – 432. 3, 78, 80, 82, 161

[62] Nagel, K., Stretz, P., Pieck, M., Leckey, S., Donnelly, R. and Barrett, C. L.: 1997, TRANSIMS traffic flow characteristics. 32

[63] Newell, G. F.: 1988, *Theory of Highway Traffic Signals*, Institute of Transportation Studies, University of California, Berkeley, CA. 3

[64] Newell, G. F.: 1998, The rolling horizon scheme of traffic signal control, *Transportation Research Part A: Policy and Practice* **32**(1), 39 – 44. 97, 103

[65] Nygaard, K.: 1986, Basic concepts in object oriented programming, *SIGPLAN Not.* **21**(10), 128 – 132. 49, 153

[66] *OmniTRANS StreamLine*: n.d.
URL: *http://www.omnitrans-international.com/en/products/omnitrans/streamline* 32

[67] Pallottino, L., Scordio, V. G., Bicchi, A. and Frazzoli, E.: 2007, Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems, *IEEE Transactions on Robotics* **23**(6), 1170 – 1183. 132

[68] Papageorgiou, M.: 1998, Some remarks on macroscopic traffic flow modelling, *Transportation Research Part A: Policy and Practice* **32**(5), 323 – 329. 86

[69] *PTV Vision*: n.d.
URL: *http://vision-traffic.ptvgroup.com/en-uk/* 32

[70] Robertson, D. I.: 1969, Transyt: A traffic network study tool, *Report RL-253*, Road Research Laboratory, Crowthorne, Berkshire, England. 38

[71] Rubinstein, R. Y.: 1981, *Simulation and the Monte Carlo method*, Wiley Series in Probability and Statistics, chapter "Monte Carlo integration and variance reduction techniques", pp. 114 – 157. 57

[72] Simonetto, A. and Keviczky, T.: 2009, Recent Developments in Distributed Particle Filtering: Towards Fast and Accurate Algorithms, *Proceedings of the* 1^{st} *IFAC Workshop on Estimation and Control of Networked Systems*, pp. 138 – 143. 165

[73] Takahashi, S., Nakamura, H., Kazama, H. and Fujikura, T.: 2002, Genetic algorithm approach for adaptive offset optimization for the fluctuation of traffic flow, *Proceedings of the* 5^{th} *International IEEE Conference on Intelligent Transportation Systems.*, pp. 768 – 772. 32

[74] *TransCad*: n.d.
URL: *http://www.caliper.com/* 32

[75] Tveit, O.: 2002, Common cycle time is a strength or a barrier in traffic light signalling?, *Traffic engineering and control* **44**, 19 – 21. 83

[76] van den Berg, J., Guy, S. J., Lin, M. and Manocha, D.: 2011, *Springer Tracts in Advanced Robotics*, Vol. 70, Springer, chapter "Reciprocal n-Body Collision Avoidance", pp. 3 – 19. 132

[77] Vigos, G., Papageorgiou, M. and Wang, Y.: 2008, Real-time estimation of vehicle-count within signalized links, *Transportation Research Part C: Emerging Technologies* **16**(1), 18 – 35. 55

[78] Webster, F. V.: 1958, Traffic signal settings, *Road Research Technical Paper* **0**(39). 77

[79] Webster, F. V. and Cobbe, B. M.: 1966, Traffic signals, *Road Research Technical Paper* **0**(56). 77

[80] West, M.: 2005, Modelling with mixtures., *Bayesian Statistics 4.Oxford: Oxford University Press (OUP)*. 56

[81] Xie, X., Barlow, G., Smith, S. and Rubinstein, Z.: 2011, Platoon-Based Self-Scheduling for Real-Time Traffic Signal Control, *Proceedings of the* 14^{th} *International IEEE Conference on Intelligent Transportation Systems, Washington DC, October, 2011*. 78, 84

[82] Yperman, I.: 2006, *The Link Transmission Model for Dynamic Network Loading*, PhD thesis, Katholieke Universiteit Leuven. 32

About the author

Nicolae Marinică was born on December 6, 1981 in Râmnicu Vâlcea, Romania. In 2005, Nicolae obtained a B.Sc. degree in automatic control and industrial informatics from University Politehnica of Bucharest, Romania. Early the same year, he spent 5 months as Erasmus exchange student at Katholieke Hogeschool Sint-Lieven Gent, Belgium. His B.Sc. thesis was focused on the development of an embedded board based on the Intel 8052 microcontroller used for a wide range of academic and commercial applications.

Nicolae completed the "Advanced Microelectronics" M.Sc. program in 2007 at the Faculty of Electronics, Telecommunications and Information Technology within the same university. The program provided knowledge for the analysis, design and evaluation of analogue and mixed signal circuits, systems and microsystems oriented to the automotive industry. Since at the time he was employed by TTTech Development Romania, the M.Sc. thesis work was done in collaboration with the company. His thesis (supervised by prof.dr.ing. Mircea Bodea and ing. Leonard Gagea) presents the implementation of two TTP communication controllers (C2NF) using an Altera FPGA.

Since November 2008, he has been employed as a Ph.D.-candidate at the SYSTeMS research group at Ghent University, Belgium, under the supervision of prof.dr.ir. Rene Boel and prof.dr.ir. Alain Sarlette. During this period, he participated different summer schools, colloquia and courses organized by Ghent University and external institutions. As part of the 8 academic partners of the C4C-EU.ICT-2007.3.7.(c) European project, his research aimed at developing control strategies for the coordination of urban traffic networks and automated guided vehicles. The thesis contributions include conception, development and implementation (in Matlab, Java) of the solutions. The Ph.D. has led so far to 4 conference papers and one A1 journal paper.